An Executive's Guide to Information Technology

Principles, Business Models and Terminology

ROBERT PLANT
and STEPHEN MURRELL

CAMBRIDGE

CAMBRIDGE www.cambridge.org/9780521853361

This page intentionally left blank

An Executive's Guide to Information Technology

Assessing the most valuable information technology for an organization is becoming a growing challenge for business professionals, who are confronted with an expanding array of options. This book is an A–Z compendium of technological terms written for the non-technical executive, allowing quick identification of what the term is and why it is significant. This is more than a dictionary. It is a concise review of the most important aspects of information technology from a business perspective: the major advantages, disadvantages, and business value propositions of each technological term are discussed, sources for further reading are provided, and there is cross-referencing with other terms where applicable. The essential elements of each concept are covered in a succinct manner so that the reader can quickly obtain the required knowledge without wading through exhaustive descriptions. With over 200 terms, this is a valuable reference for non- and semi-technical managers, executives, and graduate students in business and technology management.

Robert Plant obtained his Ph.D. in Computer Science at the University of Liverpool in 1987. He is currently an associate professor for the School of Business Administration at the University of Miami, and specializes in teaching MIS Strategy both there and at other universities and companies. His research interests focus on the role of information systems in strategic management.

Stephen Murrell obtained his D.Phil. in Computation in 1986 from the Oxford University's Programming Research Group. He is currently a lecturer in Computer Engineering at the University of Miami, where he specializes in teaching programming, algorithms, and operating systems. His primary area of research is in programming languages.

An Executive's Guide to Information Technology:

Principles, Business Models, and Terminology

Robert Plant
and **Stephen Murrell**
University of Miami, Florida

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK
Published in the United States of America by Cambridge University Press, New York
www.cambridge.org
Information on this title: www.cambridge.org/9780521853361

© R. Plant and S. Murrell 2007

This publication is in copyright. Subject to statutory exception and to the provision of relevant collective licensing agreements, no reproduction of any part may take place without the written permission of Cambridge University Press.

First published in print format 2007

ISBN-13 978-0-511-27939-3 eBook (NetLibrary)
ISBN-10 0-511-27939-6 eBook (NetLibrary)

ISBN-13 978-0-521-85336-1 hardback
ISBN-10 0-521-85336-2 hardback

Cambridge University Press has no responsibility for the persistence or accuracy of urls for external or third-party internet websites referred to in this publication, and does not guarantee that any content on such websites is, or will remain, accurate or appropriate.

Contents

Introduction

In writing this book, we have drawn upon our experiences as professors, consultants, and technologists to provide a resource for executives, students, and other readers who have a desire to achieve a rapid understanding of a technology, a computer-related process methodology, or a technology-related law.

The book provides not only the definitions for over 200 terms, but also a concise overview of the term, the associated business value proposition, and a summary of the positive and negative aspects of the technology or processes underlying the term.

The book addresses a problem faced by many executives working in the twenty-first century organization, that of understanding technology without needing to become a technologist. Today's executives use or are responsible for technology in nearly every aspect of their organization's operations; however, the pace of change in technology, the misinformation provided from informal sources, and general opaqueness of the terminology can be off-putting for executives and managers. In order to help executives overcome these problems, we have drawn upon over twenty years of teaching at the executive level and our backgrounds as technologists to provide clear, understandable descriptions for the most important technology and terminology in use within today's organizations.

The executives' need to understand technology is undeniable, but their role dictates that they must understand technology at two levels. Firstly, they must understand what the technology actually is, and this is addressed in our text by the provision of an overview section for each term. This section aims to cut through the "technobabble" and the "spin" and to provide a solid basis of understanding the technology, such that executives can appraise the role of a technology within the IT organization and the organization as a whole.

A second aspect of an executive's role is to understand the business value proposition behind the technologies, and this is addressed in the second section presented for every term. This enables executives to come up to speed quickly on the major issues and how a technology fits into the larger role it plays in their enterprise and beyond; again references are provided to enable further understanding to be gained.

A third aspect of today's executives' function is to understand their obligations with respect to legislation and regulatory frameworks such as Sarbanes–Oxley and the HIPAA acts. The book addresses this by including a set of UK and US legislative requirements under which organizations and their executives in those jurisdictions must operate.

Finally, each item is concluded with concise summaries of the positive and negative aspects generally associated with the technology. This allows executives to ascertain very quickly the key points associated with the technology, law, or process model.

While no book on technology can be exhaustive, we have aimed to provide clear and technically sound descriptions for as large a set of the terms used by technologists, vendors, and users as possible, with the aim of assisting executives and other readers in achieving as rapid an understanding of the technologies, laws, and processes as possible.

Robert Plant, Ph.D., Eur.Ing., C.Eng., FBCS
Stephen Murrell, D.Phil.

Please note that the content of all law-related articles in this book represents the views of the authors who are not legal experts, and the articles should not be read as presenting definitive legal advice.

ACM (Association for Computing

Definition: The Association for Computing Machinery was founded in 1947 to act as a professional organization focused upon the advancement of information-technology skills.

Overview

The ACM acts as a professional organization within the technology industry. The society provides a range of services, including professional development courses, scholarly journals, conferences, sponsorship of special interest groups (SIGs), and sponsorship of public awareness activities that pertain to technology, and to scientific and educational activities. The conferences, peer-reviewed publications, and SIGs provide high-quality forums for the interchange of information in the area of information technology. ACM provides an electronic portal through which a large digital library of technology-related information may be accessed, including academic and professional journals, conference proceedings, and a repository of algorithms.

Business value proposition

The ACM provides a range of resources that can assist technology professionals in developing their skills. These include continuing education programs, a high-quality database of technology literature, professional conferences, and networking events. The peer-reviewed literature includes an extensive library of algorithms allowing technology professionals to use tested and proven algorithms rather than having to reinvent them. The ACM also provides ethical guidelines for its membership and monitors public policy issues in the area of technology and science. Access to the resources of the ACM provides an IT professional with a quick, effective, and efficient mechanism through which the continuously changing fields of computer science and information technologies can be monitored.

Summary of positive issues

The ACM provides a large set of high-quality resources for the technology community. They promote ethical standards for the IT community and provide resources to assist in the governance issues of IT organizations.

Summary of potentially negative issues

Membership of a professional organization such as the ACM is optional, not compulsory as is the case in the older professions of medicine (e.g., American Medical Association, British Medical Council), law (e.g., the American Bar Association), and traditional engineering. There is no equivalent sanction to being "disbarred" for computing professionals.

References

- http://acm.org
- ACM, 1515 Broadway, 17th Floor, New York, NY 10036, USA.

Associated terminology: AIS, BCS, IEEE, W3C.

Advertising

Foundation concepts: Web, e-Commerce.
Definition: Web-based advertising covers many techniques and technologies, each of which is intended to attract the user to take notice of the advertising and, where appropriate, act upon that advert.

Overview

Advertising via the internet has been growing in popularity and sophistication since the internet was deregulated in 1995 and has spawned a series of advertising initiatives. An early form of advertising was the banner ad, a simple block at the top of a web page announcing a service or product;

when the user clicked upon it, it acted as a hyperlink taking them to the new site and presenting more information on the product, and, of course, an opportunity to buy it. The advertiser would then pay the original web site for the traffic that "clicked through," and possibly pay more should a successful transaction occur.

Banner ads proved in many cases to be less effective than desired and so the industry started to create personalization software based upon individuals' online web-surfing activities, the goal being to present specific products and services based upon users' past preferences. These personalization systems did draw criticism from civil liberties groups concerned that they could profile individuals, and selling the information would be an infringement of the right to personal privacy.

As the internet evolved, more sophisticated advertising developed, including the infamous *Pop-up ad*, an intrusive message that suddenly appears upon the screen in its own window. Pop-ups are often embedded Java applets or JavaScript programs that are initiated from the web page as it is viewed; usually the pop-up window can be closed, but some of the more persistent pop-up advertising cascades so that when one window is closed another opens, slowly taking over the machine's resources. It may be necessary to shut the machine down completely to escape this situation. Careful control of the web browser's security settings can usually prevent the problem from occurring in the first place.

A form of advertising that is also intrusive, and in some instances criminal, is *Adware* (the term is used to denote a certain type of intrusive advertising but is also the registered and trademarked name of a company that sells anti-spyware and anti-spam software). Adware takes the form of pop-up advertisements or banner ads and comes from several sources. One source is certain "freeware" or "shareware" products that vendors give away without charge but have embedded advertising into the code so that it suddenly appears upon the screen and may or may not be removable, or could be subject to a time-delayed closure. As a condition of use, some adware programs go on to install programs on the user's machine, known as *Adbots*, and these programs then act as a type of *Spyware* sending back information to their advertising source pertaining to the user's behavior. Often these additional installations are undisclosed.

Illicitly installed software may redirect web browsers to access all pages through a proxy, which can add banner advertisements to totally non-commercial pages that in reality bear no such content. Home-page hijacking is also a common problem: a web browser's default or home page may be reset to a commercial site, so that, every time the web browser is started up, it shows advertisements or installs additional adware.

Business value proposition

Well-placed internet advertising can be of significant benefit to the company or individual sending it out. In the 2000 US presidential election the Republican Party's use of email to spur its registered voters in the State of Florida to go out and vote has been acknowledged as influential in George W. Bush's closely contested victory over Al Gore, and changed the nature of political campaigning. The internet can also be used as a non-intrusive distribution model for advertising that is driven by user demand, an example of which was the advertising campaign by BMW, which used top film directors, writers, and famous actors to create short films in which their vehicles were as much the stars as the actors. The "BMW films" series proved to be hugely popular and reinforced the brand.

Many early dot-com-era (1995–2000) companies attempted to finance their business

through the use of a *Click Through* advertising model, but were unsuccessful since the revenues generated were in most cases insufficient to sustain the business. Online advertising has been used effectively by some companies, a prime example being Google, as a revenue source aligned to a successful business model and popular product.

Many companies now offer pop-up blockers, spam blockers, and other anti-advertising software, and hence the ability of advertisers to reach their audience legally has been curtailed. With the continuous development of new technologies, the opportunity to create new advertising media continues to develop in tandem.

Summary of positive issues

Online advertising is a relatively low-cost mechanism for the dissemination of commercial messages. The basic technology behind internet advertising is well known and continues to evolve, and consequently fresh approaches are constantly being developed. Non-intrusive internet advertising, where consumers pull the advertising rather than the advertising being pushed onto them, can be very successful. Forms of advertising that don't work will (with any luck) very quickly die out.

Summary of potentially negative issues

Some forms of advertising are illegal, spam for example. Internet advertising is frequently thought of as annoying and intrusive by the recipient. Anti-advertisement software and systems are continually being improved to block intrusive advertising. It is difficult to imagine why any customer would do business with a company that took over their computer with cascading pop-up advertisements, and corporations would be well advised to think very carefully before embarking upon such a campaign.

Reference

- R. L. Zeff (1999). *Advertising on the Internet* (New York, John Wiley and Sons).

Associated terminology: Internet, JavaScript, Spam, Spyware, Web browser.

Agent

Foundation concepts: Artificial intelligence.
Definition: A self-contained software component acting on the behalf of a human controller.

Overview

An agent, sometimes also called a "bot," is a software application intended to perform some service for a human controller without requiring constant supervision. A simple application could be to carry out a particular bidding strategy in an online auction. After being programmed with the human controller's desires, an agent could constantly monitor the state of the auction, reacting to competing bids as instructed, without any intervention; the human controller would be free to get on with life in the meantime.

Another valuable application of agents, that is only just entering the realms of possibility, is in information gathering and filtering. The internet provides a vast quantity of news and information sources, constantly being renewed and updated. It is impossible for a human reader to monitor even a small fraction of it all; even after identifying a moderately sized set of sources for a particular area of interest, keeping up to date with new developments requires the devotion of significant periods of time. If an agent were programmed to recognize articles that would be of interest, it could maintain a permanent monitoring presence, filtering out just the few items of genuine interest, and presenting them to the human reader on demand. This use of agent technology could provide

a major increase in personal productivity, but requires that the *natural language* problem be mostly solved first. The problem is that software must be capable of fully understanding human prose, not just recognizing the words, but determining the intended semantics, and that is still the subject of much current research.

It is generally envisaged that agents will not be static presences on the human controller's own computer, but should be able to migrate through the network. This would enable an agent to run on the computer that contains the data it is reading, and would therefore make much better use of network bandwidth. It does, however, introduce a serious security and cost problem: agents must have some guarantee of harmlessness before they would be allowed to run remotely, and the expense of providing the computational resources required would have to be borne by somebody.

The idea of an agent as a meaningful representative of a person, acting on their behalf in business or personal transactions, is still very much a matter for science fiction. The problem of artificial intelligence, providing software with the ability to interact with humans in a human-like manner, is far from being solved. It is believed by some that a large number of relatively simple agents, in a hierarchy involving higher-level control agents to coordinate and combine efforts, may be the best way to create an artificial intelligence. This theory remains unproven.

Business value proposition

The use of agents in business has some degree of controversy attached to it. Simplistic agents have been used for some time to carry out stock market transactions. If it is desired to sell a particular stock as soon as it crosses a particular price threshold, constantly monitoring prices to catch exactly the right moment would require a lot of human effort. This is exactly the kind of condition that can easily and reliably be detected and acted upon by very simple software. However, it is widely believed that the extreme speed of these agents has a destabilizing effect on the markets, and may even have been responsible for some minor market crashes. The slowness of human reactions, and the need for a real person to make orders, allows common sense to slip in where it could not be provided by a simple automatic agent.

Agents are also used by some online financial services such as automobile-insurance consolidators, who send their bots out to the sites of other companies. The bots fill out the forms on those companies, retrieve a quote and then these quotes are all displayed on the consolidator's site. The unauthorized use of bots on third-party sites is prohibited by many organizations; however, preventing their access to a web site open on the internet can be almost impossible.

Summary of positive issues

Agents can be created and used to automate processes. Agents can be sent out over a network to collect and return information to their owner.

Summary of potentially negative issues

Agent technology is primitive and only well-structured relatively simplistic tasks can be performed safely. Bots and agents are very unwelcome on many web sites and networks, and web site providers should be aware that other organizations may make covert use of their online resources, presenting the results as their own; that can result in an unexpected increase in bandwidth and server load when a popular service is provided.

Reference

- J. Bradshaw (1997). *Software Agents* (Cambridge, MA, MIT Press).

Associated terminology: Artificial intelligence, Natural language processing.

AIS (Association for Information

Definition: The Association for Information Systems (AIS) was founded in 1994 and acts as a professional organization for academics who specialize in information systems.

Overview

The AIS acts as a professional organization within the technology industry. The aim of the organization is "*to advance knowledge in the use of information technology to improve organizational performance and individual quality of work life*." (www.aisnet.org). The society provides a range of services, including professional development courses, scholarly journals, conferences, and the sponsorship of special interest groups (SIGs). The AIS is primarily focused on the role of information systems in a business context, and is closely aligned to servicing the needs of management information systems (MIS) professionals and academics who study the field of MIS. The AIS, which merged with ISWORLD (*Information Systems World*) in 1998, also merged its major conference, the AIS Conference on Information Systems, with the highly respected International Conference on Information Systems (ICIS) in 2001. ICIS provides a forum for the presentation of high-quality papers and acts as the major annual academic recruiting conference for MIS faculty.

Business value proposition

The AIS provides MIS professionals with a set of resources through which their professional skills may be developed. These resources range from continuing education courses and access to a high-quality database of technology literature to professional conferences.

The AIS provides a link between academic research and industrial practice within the field of management information systems. The society publishes two peer-reviewed journals, the *Journal of AIS* and *Communications of AIS*, and has over 20 specialist groups, including IT in Healthcare, Enterprise Systems, E-Business, and Accounting Information Systems. The specialty groups provide forums through which academic–industrial research collaborations are undertaken.

Summary of positive issues

The AIS provides a set of high-quality resources for academics and practitioners within the domain of management information systems.

Summary of potentially negative issues

Membership of a professional organization such as the AIS is optional, not compulsory as is the case in the older professions of medicine (e.g., American Medical Association, British Medical Council) and law (e.g., the American Bar Association).

References

- http://www.aisnet.org/.
- Association for Information Systems, P.O. Box 2712, Atlanta, GA 30301-2712, USA.

Associated terminology: ACM, BCS, IEEE, W3C.

Algorithm

Foundation concepts: Program, Formal methods.
Definition: A set of explicit, unambiguous instructions for solving a problem or achieving a computational result.

Overview

An algorithm is, in a way, a solution to a problem. It is a complex of instructions, which, if followed exactly, will produce a particular result. The method for long multiplication as taught in elementary schools is a good example of an algorithm. The overall task, multiplying together two large numbers, is not something that a person can normally do in one step; we need to

be taught a method. The particular way of performing the task, running through the digits of the multiplier one by one, multiplying each by every digit of the multiplicand, writing the resultant digits in a particular pattern, and adding them all up in columns at the end, is an algorithm for multiplication.

Since computers are simply devices that are good at obeying explicit unambiguous sequences of instructions, algorithms are the logical constructs that programs embody. One of the key tasks of programming a solution to a problem is designing a good algorithm.

Just as there are many different ways to solve problems in real life, there are often many different algorithms available to a programmer for one task. An office worker who has to alphabetize a pile of forms may quickly fall into a particular pattern, and get through the job almost mechanically.

(1) Start with all the forms in a pile on the left, and an empty space for a new pile on the right.
(2) Pick up the first form and set it aside.
(3) Scan through all of the remaining forms in the first pile, comparing each with the one that was set aside; when you find a form in the left pile which belongs after the one set aside (in alphabetical order) then swap them: set aside the one from the pile, and put the previously set-aside one back in the pile.
(4) Put the set-aside form on top of the new (right) pile.
(5) If the original (left) pile is not empty, go back to step (2) and repeat.

This is a sequence of instructions that could easily be followed by a worker who has no idea how to sort forms, or, indeed, no idea of what "sorted" means. Without requiring any intelligence, this procedure allows anyone to perform the task.

Of course, it is necessary to know how to "pick up" a form, and how to compare two forms to see which comes first in alphabetical order. Office workers know how to perform these activities, but it is possible to produce an even more detailed algorithm for these subtasks too.

That is the nature of an algorithm. Every required action must be spelled out, so that the task may be performed by a worker who has no understanding of it. At some point it must be assumed that the worker knows *something*. They must be capable of following instructions for example, so algorithms do not regress into an infinite fuzz of infinitesimal detail. Even computers already know how to do some things: they can add and subtract and follow instructions. An algorithm for computer execution simply instructs how to perform a task in terms of subtasks that the computer already knows.

The five-step sorting algorithm detailed above is a realistic one that people do use, and it works. Unfortunately, it is satisfactory only for small piles of forms. If you start with just ten forms, the first is set aside, and compared with all nine of the others, to find the one that goes on top of the second pile. Then, the first of the remaining nine is set aside and compared with all eight of the others, to find the next. Then the first of the remainder is compared with all seven of the rest, and so on and so on. By the end of the procedure 45 individual comparisons will have been needed, which does not seem so bad. However, if you started with 20 forms in the pile, 190 individual comparisons would be needed: twice the number of forms, but four times the work. For large piles of forms, this algorithm can be impossibly slow. A mere ten thousand forms would require nearly fifty million comparisons.

It would be an absolute disaster if programmers had to rediscover this well-known fact every time a programming task

called for inputs to be sorted. Repeated experience tells us that most programmers do not notice that the simple sorting method is too slow for large data sets, and, of those who do notice, very few indeed are able to work out a significantly faster alternative. One of the essential parts of a formal training in programming is a long and demanding study of the large collection of algorithms that have already been discovered and analyzed, together with the *Data Structures* (carefully tailored, seemingly unnatural ways of organizing data for effective access) that go with them. As with any other engineering profession, it is impossible to do a good job without a thorough knowledge of what has been tried before. If a programmer starts the job fully armed with what is already known, they will have some chance of finding something new. Inventiveness is important: not all problems have been seen before. A programmer who does not already know the standard algorithms and data structures is doomed to nothing more than rediscovering the basics.

Business value proposition

Many of the algorithms and data structures for standard and interesting problems are thoroughly documented and analyzed in sources such as those provided by ACM, and the standard textbooks well known to all trained programmers. Knowledge of established algorithms not only gives programmers the tools necessary for solving standard problems without having to "reinvent the wheel" at every step, but also provides an essential foundation for designing truly new solutions. As Isaac Newton in 1676 explained his success in inventing calculus and understanding gravity, "If I have seen further it is by standing on the shoulders of giants."

Algorithms are an abstraction, a high-level view of a method, that enables programmers to construct and investigate a design to solve the computational problem independently of the final implementation language. This description may be written in a style that is easier to read than finished program code, and thus amendments to the design can be made in a more informed manner than would be possible by examining the code. Knowledge of algorithms and associated data structures also enables programmers to determine the *efficiency* of their solutions. A simple and obvious algorithm that works well in simple cases may be completely unsuitable for commercial use. A professional programmer must thus understand that alternate algorithms exist, and be able to select the appropriate one, perhaps even invent a new one, for any given situation. Further, the *computability* of an algorithm must also be considered, to ensure that the algorithm is theoretically sound.

Summary of positive issues

Algorithms are an abstraction of a problem to be solved; thinking of a general algorithm rather than a specific problem allows a programmer to write cleaner, more maintainable code that has a strong chance of being reusable in other projects. An enormous collection of existing fully analyzed algorithms with well-tested implementations is freely available, and a vast range of books is also available on the subject. Design, analysis, and proof of new algorithms continues to be a major direction of research.

Summary of potentially negative issues

Algorithmic design needs qualified specialists in computer science and software engineering. Inappropriate algorithm design can result in inefficient or unworkable solutions being implemented. Lack of knowledge of algorithmic techniques by software professionals is a major cause of inefficiency and poor performance.

References

- N. Wirth (1978). *Algorithms plus Data Structures Equals Programs* (Englewood Cliffs, NJ, Prentice-Hall).
- D. E. Knuth (1975). *Fundamental Algorithms* (New York, Addison-Wesley).
- M. Hofi (1975). *Analysis of Algorithms* (Oxford, Oxford University Press).

Associated terminology: Efficiency, Computability, Programming language.

Analog

Definition: An analog system is one that operates on a continuous scale.

Overview

Most real-world information is not directly perceived in digital form. Sound is really a complex and continuously varying pressure wave in the air, which we "hear" when it impinges upon small hair-like sensors in our ears, setting them aflutter. A visual look at the world reveals an infinitely detailed spectrum of shapes, colors, sizes, and textures. Even simple measurements such as the length of a piece of string or the weight of a potato have unlimited precision: if a string is stated to be 293 mm long, clearly that is just an approximation limited by the ruler used; more careful measurement might reveal it to be 293.1, or 293.097 632 63 mm long.

The contrast between analog and digital is that analog measurements are thought of as corresponding naturally to real-world measurements and having unlimited natural precision, whereas digital measurements are always encoded as a string of digits and are therefore limited in precision by the number of digits written.

[Side note: Modern physics is of the opinion that the Universe actually has a fundamentally digital nature: light travels in particles called quanta, space is somehow granular and can not be infinitely divided, even time may really tick rather than flowing continuously. All of these phenomena occur on a scale far too small to be relevant in normal life, and some are still quite controversial even regarding their supposed existence.]

Traditional electronics built with resistors, transistors, vacuum tubes, etc. also has a fundamentally analog nature. A voltage is just as infinitely variable as the length of a piece of string. Before the digital age, electronic and even mechanical analog computers were common. In an electronic analog computer, all physical quantities, whether they are lengths, weights, or anything, are directly represented by a related electronic quantity: voltage, current flow, etc. A device that can add voltages (e.g. inputs of 3.1 V and 2.6 V producing a single output of 5.7 V) can be used to add lengths or weights. Even into the 1970s tide tables were still being created by analog computers.

Analog computers have inherent problems: controlling voltages and currents to a very precise degree is exceptionally difficult and expensive; long-term storage of accurate analog data is practically impossible. Now, everything that could be done by analog computation can be done more accurately and reliably by encoding the analog inputs in digital form and performing a digital computation on it.

The only concession made by a modern computer to the analog "real world" is usually a few analog–digital converters. The term "A to D," or A/D, is used for a device that takes an analog input (such as sound or voltage) and encodes it in digital form. The term "D to A," or D/A, is the reverse. A computer with a microphone has a simple A/D device to convert the sound into digital form. A computer with a loudspeaker has a simple D/A device to convert digital signals into audible form.

Business value proposition

Analog computing has largely been marginalized by the digital age. Except for very small systems, nearly everything that could be done by an analog system can be done

more easily and accurately with a digital system. The analog world does not generally impact modern business computing.

Summary of positive issues

Analog processing is a requirement when interfacing a computer system with real-world inputs and outputs.

Summary of positive issues

Analog processing is inherently inaccurate and inconsistent, but all analog signals may be converted to digital form before processing, thus negating such issues.

Associated terminology: Digital, Audio, Video.

Anti-virus software

Foundation concepts: Virus, Security
Definition: An application that detects and removes viruses and other illicit software.

Overview

Anti-virus software has the ability to recognize viruses that have already been investigated by the software authors. Generally, a "signature" for each known virus is created, which consists of certain recognizable key attributes. This means that slightly modified viruses, or viruses embedded within other files, will also be detectable.

A virus, beyond consisting of executable code, may have any form. There is nothing about a virus *per se* that can be detected. Only already discovered forms may be detected. Fortunately, most viruses are unimaginative copies of existent forms, and anti-virus software can be extremely effective.

However, when a truly new virus is released upon the world, there is nothing that any anti-virus software could possibly do to detect it. Protection relies upon the authors of the anti-virus software continually monitoring thousands of vulnerable systems, watching for unexpected activity, and catching new viruses as quickly as possible. Once a new virus has been caught, a new signature can be generated, and uploaded to all operational anti-virus software.

It is essential that anti-virus software be continually updated. Reputable anti-virus producers provide frequent automatic updates through the internet. Anti-virus software can only detect a virus that was already known at the time of its last update. There is no protection against a new form of virus.

When anti-virus software detects a virus, it will generally attempt to remove it from the system cleanly, restoring affected files to their proper state. Sometimes this repair operation is not possible, and the last resort is to quarantine the damaged file. This means simply moving it to a protected location from whence it can not be executed, thus rendering it harmless without destroying the potentially important file that it is attached to.

Anti-virus software generally works in two modes. Periodically it begins a "scan," actively inspecting every single file on disk or in memory, looking for signs of infection. This is a long process, and is typically configured to happen overnight. In addition to periodic scans, the software will also inspect all incoming (and possibly outgoing) data, including emails, downloaded web content, inserted floppy disks, etc. Anti-virus software can also be configured to monitor currently running software, as well as that resident on disk, and halt any active viral processes.

Business value proposition

Anti-virus software is a vital aspect of a corporate security framework. A plan must be in place to ensure that the latest anti-virus software is active and loaded onto all email servers to intercept emails with known viruses and take the appropriate action. Anti-virus software should also be

present upon client computers to intercept infected software that is brought to that computer through downloads, CDs, and other portable memory devices. The use of self-updating software is the recommended approach since this reduces the possibility of human error in maintaining the anti-virus software.

Summary of positive issues

Anti-virus software is easily accessible, deployable, and maintainable. Reputable vendors actively work to identify new and mutated viruses and maintain their software to defend against new threats.

Summary of potentially negative issues

New viruses are continually being developed and deployed, therefore all new viruses need to be identified and new defenses created and built into the anti-virus software. However, until the anti-virus software updates have been received and (automatically) installed, there remains a degree of vulnerability.

Reference

- P. Szor (2005). *The Art of Computer Virus Research and Defense* (New York, Addison-Wesley Professional).

Associated terminology: Spam, Trojan horse, Advertising, Spyware.

Application development methods

Foundation concepts: Software development.
Definition: An application development method is a process model through which a software program or system is developed.

Overview

The development of individual software programs and larger software systems is, except for trivial problems, a complex task. Thus, in order to help application developers and project managers cope with this complexity, attempts have been made to create process guidelines or "development methods" through which systems can be created. These methods structure the development as a set of smaller stages or steps that focus upon specific issues usually encountered at a specific stage in development. The models act to ensure quality levels and documentation levels, and to help managers develop their project-management plans.

The range of application development methods mirrors the needs of the user communities in which they are employed. The most rigorous of these are the *Formal methods*, which are based upon the use of mathematical notations and use proof systems to ensure that the implementation matches the specification. Commercial methods include *RAD* (Rapid Application Development) and *JAD* (Joint Application Development). RAD is an approach based upon prototyping and was developed to facilitate team software building, while JAD is a methodology based upon the use of focused meetings through which systems are developed. Traditional models of development include the *Waterfall model*, a stage-based lifecycle model used since the 1970s to develop business applications. The development of *Object-oriented* software in the 1980s led to methods that assist in the development of object-based systems; these range across the formality spectrum from formal methods such as Object-Z to the more standard commercial approach known as Object-Oriented Analysis and Design (OOAD).

Methodologies have also been devised to help developers with systems development in specialized areas such as knowledge-based systems, neural networks, graphical user interfaces, parallel processing, and web development.

Business value proposition

The role of applications methodologies is to facilitate the development of higher-quality systems with the minimum of effort. The

employment of specific methods enables standards to be enforced across the development effort.

Summary of positive issues

A wide range of methods are available and some knowledge of the possibilities helps developers select the most appropriate method for the task to be performed. Use of the appropriate methodology engenders higher-quality software development, faster development, and heightened maintainability over the life of the system.

Summary of potentially negative issues

The wide range of methods available requires the developer to select the most appropriate method with care. Some, such as *Formal methods*, require specialist training and high degrees of mathematical knowledge and experience on the part of the developer. Some methods are "weak" and, rather than being based upon mathematical proof of correctness, rely upon a testing strategy for measuring their correctness. Some methods have more tools available to the developers using them than others. Each of these factors needs to be understood for the method being deployed since they significantly impact the quality of the system developed, as well as the time, effort, and resources necessary to develop software. Failure to select the correct method can have significant negative consequences on the development process.

References

- I. M. Holland and K. J. Lieberherr (1996). "Object-oriented design," *A.C.M. Computing Surveys*, Volume 28, Issue 1.
- E. Clarke and J. M. Wing (1996). "Formal methods: state of the art and future directions," *A.C.M. Computing Survey*, Volume 28, Issue 4.
- R. Plant and R. Gamble (2003). "Methodologies for the development of knowledge-based systems: 1982–2002," *Knowledge Engineering Review*, Volume 18, Issue 1.

Associated terminology: Formal methods, Object-oriented, Waterfall model, Joint application design, Rapid application development.

Application generator

Definition: An application generator is a tool that aids in the generation of executable or source code from a high-level design or model.

Overview

The term *Application generator* grew out of the older term *Code generator*, the name of the part of a *Compiler* that actually produces the final executable code. The notion of a program developing a solution from a higher-level description then evolved throughout the 1980s into the area of *Computer-aided software engineering* (CASE), which focused upon the generation of programs from high-level modeling environments that developers manipulated through graphical user interfaces. The CASE initiatives ranged from very large systems built by major vendors that attempted to cover the entire software development lifecycle to smaller systems that provided tools to enable developers to build and link reusable software modules or libraries. The 1980s and 1990s saw an explosion of interest in the *Object-oriented* paradigm of software development and efforts to capture this market resulted in application generators specifically aimed at the development of object systems.

Vendors have built a wide range of application generators that support a variety of development methods, models, and applications, including *Rapid application development*, *UML*, *Object-oriented* models, *Formal methods* such as *B*, and *Structured analysis and design*. These development methods produce results in a variety of programming

languages and systems, such as Ada, Cobol, Java, C++, C, Visual Basic, PL/1, DB2, SQL, and DDL.

Business value proposition

The use of application-development environments enables organizations and individual developers to create executable code rapidly from a high-level design. The application generators provide a uniform development environment, e.g., a UML GUI, through which code of consistent quality is generated. The code that is generated can then be tested and maintained through other functional aspects of the system. This style of development is especially suitable to situations in which repetitive incremental designs that operate upon the same database and structures need to be developed. Development through application generators allows the programmers to focus upon more important problems, testing, and add-on functionality.

Summary of positive issues

Application generators provide developers with the potential to create high-quality designs rapidly, using systems that enforce design rules through the developers' interface, from which consistent documented source code is generated. The design methods available for support range from formal methods to traditional lifecycle methods. Similarly the range of languages output by these systems is equally wide, ranging through Ada, Java, C++, Cobol, and real-time systems. Many tools are available, some from open-source vendors and others from commercial organizations, with corresponding levels of support.

Summary of potentially negative issues

Application generators are not intended to provide a complete substitute for human programmers but can be used to relieve them of repetitive, unrewarding, and error-prone programming tasks. The developer will need to be trained in the methodologies used by the systems and must understand the implications and the limitations of the code being generated. The code generated will be consistently based upon the design, but the performance of that code will need to be assessed if performance is critical, since the code is often not optimized. Furthermore, it is important to determine whether designs and code from different systems can be shared across tools, architectures, and databases.

Reference

- J. Herrington (2003). *Code Generation in Action* (Greenwich, CT, Manning Publications).

Associated terminology: UML, Object-oriented, Formal methods, Structured design methodologies, Cobol, Java, C++, C, Visual Basic.

Application server

Foundation concepts: Server, Client.
Definition: An application server is a computer upon which applications reside and that allows those applications to be accessed by client computers.

Overview

Network architectures are sometimes defined in terms of tiers, and the terms *Two-tier* and *Three-tier architecture* are frequently used. Each tier represents a layer of technology. In a two-tier architecture *Clients* typically represent one layer of the architecture, and the *Server* or servers represent the other. In three-tier architectures a middle tier is placed between the client and the server. This intermediate tier is sometimes referred to as the *Application server* and controls the information flow and directs requests that are made upon other system resources such as a *Database server*, which may be distributed across the third tier.

The three-tier model allows greater numbers of clients to be connected to the

network as the application server in the second layer works to balance the workload and optimize data use and data availability. The three-tier model became popular in the 1990s and eased the adoption of a range of network architectures, including the thin-client model.

The term *Application server* is also associated with providing services to applications based upon development using the Java 2 Platform (Enterprise Edition), known as J2EE, a system for creating component-based multi-tier enterprise applications.

Business value proposition

Application servers are used to provide optimal access to applications and resources on a network. They facilitate the adoption of thin-client network architectures and relieve the network's database servers from having to work on task and process management.

Summary of positive issues

Application servers are well supported by major software vendors. Application servers can relieve and balance the load on the primary network servers.

Summary of potentially negative issues

The use of an additional server in a network requires the expenditure of capital, physical, and personnel resources.

References

- D. Sadoski (1997). *Two Tier Software Architectures: Software Technology Roadmap* (Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University).
- D. Sadoski and S. Comella-Dora (2000). *Three Tier Software Architectures: Software Technology Roadmap* (Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University).

Associated terminology: Client–server.

Application service provider (ASP)

Foundation concepts: Internet, Server.
Definition: A third-party provider of applications that are accessed through the internet.
Not to be confused with: Active server pages, a Microsoft web product.

Overview

The concept of an ASP is based upon the concept of outsourcing. Just as an organization may purchase its security services or its janitorial services from an outsourcer, some companies have determined that they should purchase software services from a third party. The role of an ASP can be as simple as a web *Hosting* company or an *Internet service provider* where basic *Web services* are purchased, relieving the client company of the responsibility for maintaining a web site and an external internet connection. More complex application services provided by ASPs range from monthly payroll processing services in the style of a 1970s computing *Bureau*, to the provision of full *Enterprise resource planning systems* and services. ASPs operate these functions on behalf of their clients, owning and running the software applications at secure locations that provide full redundancy in terms of power sources, data backups, and remote duplicate application servers.

Business value proposition

ASPs provide companies with an alternative to developing and providing their own applications. The ASP concept has been a part of computing for a long time, with "Computer bureaux" being a central part of data processing in the 1960s and 1970s when resources such as specialist printers and payroll processing were expensive to own and run internally. The CIOs of today's organizations face the same cost of ownership and return on investment issues as their predecessors faced in the era of bureaux and also share concerns over the

control of access to proprietary information and ensuring flexibility in the solution options.

The internet has allowed universal access to applications, which, combined with reliable technology, provides a versatile and secure ASP environment. ASPs provide small businesses with an option to service their traditional processes such as internet service provision, web hosting, taxation, and human resource management (HRM), and, depending upon requirements, contracts can be obtained in a variety of ways. For example, they may be based upon transaction volumes, bandwidth requirements, and database requirements, and pay-as-you-go payment scales are also available. The ASPs provide useful functionality and maintain high skill levels for their employees, thus reducing the training and HRM requirements for those companies using them. ASPs also provide a flexible solution option for CIOs in times of organizational change or re-engineering.

The dilemma facing CIOs is that of how to balance costs and flexibility. ASPs prefer to have a fixed set of requirements and charge more for flexibility in contracts, whilst CIOs prefer high degrees of flexibility and control with low costs. Contract negotiation is thus a central component of the ASP engagement process. CIOs are also required to determine which functions are core to their business and, no matter how good the ASP, failure in an outsourced core function at an ASP could be fatal to an organization.

Summary of positive issues

The ASP model offers a third-party solution for companies and their software application requirements. ASPs can be used as backup mechanisms for ensuring continuous process operation. ASP contacts may be undertaken through a variety of payment options. Non-core processes can safely be passed to ASPs and run through the internet.

Summary of potentially negative issues

There is a trade-off between the cost associated with using an ASP and the flexibility offered by it. Core processes outsourced to an ASP are potentially at risk. Internal learning and process improvement may potentially be restricted by the running of processes through an ASP.

References

- M. Lacity and L. Willcocks (2000). *Global Information Technology Outsourcing: In Search of Business Advantage* (New York, John Wiley & Sons).
- J. Harney (2002). *Application Service Providers (ASPs): A Manager's Guide* (New York, Addison-Wesley Professional).

Associated terminology: CIO, Enterprise resource planning, Internet, Web services.

Architecture

Foundation concepts: Hardware, Software.
Definition: The overall design of a system.

Overview

The design of any system of any kind may be viewed at a variety of different levels. The top-level view, concerned with how the components fit together and interact, rather than how the individual components are constructed, is known as the system's architecture. This is a direct extension of the plain-English meaning of the word: the overall design of a building, concerned with the structure of the building as a whole, not considering the equally important but lower-level details of how bricks are made or the color of the bathroom fittings. It is perhaps telling that the word is derived directly from a Greek compound meaning "the most important craft."

Computer architecture is concerned with how the registers, arithmetic units, control components, etc. are combined to make an

efficient working CPU, not with how the registers and arithmetic units themselves are made.

Network architecture refers to the overall "map" of a network: which computers, routers, servers, switches, etc. are connected to which, in what local groupings, and through which protocols.

Software architecture is concerned with how the major components of an application, perhaps a database, a user interface, and a network service provider, are combined together to make a single, integrated, well-engineered whole.

Business value proposition

The term "architecture" is used by technologists and systems professionals in many ways: computer architecture, network architecture, software architecture. For CIOs the term "systems architecture" is frequently used to describe the overall corporate computing resource which contains as subsystems the other architectures. When designing systems architectures many issues are considered, ranging from technical issues such as the technical limitations and capabilities of various competing architectures to the ability of a particular architecture to support business processes.

The selection and implementation of an architecture can be further complicated in several ways, including by the nature of the organization's existing systems, which may or may not be compatible with the strategic and technical requirements of the envisioned architecture, and the influence of external entities such as major customers who may place pressure upon a supplier to use a particular technology, which then has to be integrated into the overall architectural design. The selection of a new architecture may be influenced by factors such as regulatory frameworks that require a certain technology to be used, or the strategic decision to use open-source software as a core of the software architecture.

The CIO and the IT organization within an enterprise must develop, document, and maintain systems architectures with the aim of supporting and enabling the business processes of that enterprise. Intimate knowledge of the architecture at all levels enables technology and business decisions to be made in such a way as to facilitate the continued alignment of corporate strategy and the technological architecture that underpins it.

Reference

- S. Spewak (1993). *Enterprise Architecture Planning: Developing a Blueprint for Data, Applications, and Technology* (New York, John Wiley and Sons).

Associated terminology: CPU, Enterprise resource planning.

Artificial intelligence (AI)

Definition: Computer-based technologies that simulate or exceed human levels of task-related performance or expertise, or that duplicate some aspect of intelligent human behavior.

Overview

The origins of artificial intelligence stem from the research of workers such as Alan Turing, who in the 1930s asked "Can a machine think?" and devised a test known as the Turing test as an understandable way to approach the question. The test basically consists of a human judge undertaking a conversation via a terminal and keyboard (as in a Telex system) with a selection of other entities. Some of those others may be real humans, others will be computer or robotic systems. If the human can not reliably tell which is which, Turing's claim is that we must accept that the non-human participants have achieved intelligence. The point of the test is to isolate judgment from irrelevant or "unfair" influences: many claim that machines can never really *think* because they have no souls, or are only

obeying their programming. The Turing test allows a purely objective assessment based on merit alone.

Since its inception, AI has developed many branches and can be thought of in a variety of ways. The branches of AI include *Machine learning*, *Natural language understanding, Neural networks*, general problem solving, *Robotics*, *Expert systems*, vision processing, and speech recognition. These sub-disciplines have evolved at varying rates, with some aspects still being confined to research laboratories, whilst others have become mainstream software design techniques.

In its modern sense, AI is not limited to attempts to duplicate or simulate intelligent human behavior. The basic techniques of AI are frequently used in solving problems for which no method of solution is known. *Heuristic* and *Minimax* searching allow a computer application to search effectively an infinite range of possible solutions and find one that is at least adequate. *Neural networks* can learn to extrapolate solutions to problems that have never arisen before by self-training based on samples of similar problems and their known solutions. These and other techniques follow the human pattern of learning from and adapting past experiences; they can not be relied upon to produce perfect or optimal solutions, but when no *Algorithm* is known, and no programmer has been capable of creating a direct solution, they may be the only alternative available.

Business value proposition

Artificial intelligence covers a wide variety of areas, and the formulation of a working solution frequently requires the incorporation of a wide variety of disciplines, techniques, and skills. While this can be an extremely difficult task to manage and incorporate into a working application, the incorporation of these AI techniques frequently provides the only possible solutions, traditional approaches being unsuitable or nonexistent. The incorporation of AI solutions into an application can therefore result in a distinctive differentiated solution for an organization's products, for example washing machines that incorporate fuzzy logic may be superior in performance to traditionally controlled washing machines since they may be able to adapt better to unforeseen combinations of circumstances.

Summary of positive issues

Artificial intelligence has matured markedly since its early days, and many of the early techniques and applications that were experimental and cutting-edge have subsequently become part of a computer scientist's general "tool kit." For example, the development of expert "knowledge-based" systems may now be carried out through mature established commercial applications and supported through well-researched development models and testing techniques. Artificial intelligence tends to refer to methods that are still in the research arena, and when the technology becomes mature it merges into the general set of techniques available to software engineers.

Summary of potentially negative issues

The newer techniques and applications available through this branch of computer science are advanced in nature and require specialized knowledge to design, implement, and test. Consequently they are expensive to develop. Additionally, the research nature of many aspects of AI requires careful management consideration before adoption, as does the suitability of AI systems in respect to high-risk implementations. The validation and verification of AI systems requires high degrees of specialized knowledge and remains an area of active research.

It is essential to be aware of the true nature of any AI technique built into a product; many produce generally very good results, but with no guarantees, and scenarios at the edge of a system's range may produce markedly poorer results than customers expect.

Reference

A. Barr and E. Feigenbaum (eds.) (1981). *The Handbook of A.I.*, Volumes I, II, and III (London, Pitman).

Associated terminology: Application generator, Expert systems, Fuzzy logic, Machine learning, Natural-language processing, Robotics, Algorithm.

ASCII

Foundation concepts: Digital, Bit.
Definition: American Standard Code for Information Interchange: the standard encoding for text stored on computers.

Overview

All information stored on a computer has to be in numeric form. ASCII provides a numeric representation for every symbol that can be typed on a "standard" keyboard. Each character is represented by a unique small number as follows:

0–31	invisible characters such as ENTER and TAB.
32	space
33–47	! " # & % $ ' ( ) * + , - . /
48–57	the digits 0 1 2 3 4 5 6 7 8 9
58–64	: ; < = > ? @
65–90	the capital letters A B C D E F G . . . X Y Z
91–96	[\ ] ^ –
97–122	the lower-case letters a b c d e f g . . . x y z
123–126	{ | } ~

thus any typed text may be represented by a sequence of small numbers.

This code is still used as the primary representation in nearly all computers, but it is rather primitive in concept. Providing representations only for the keys that appeared on a teletype, ASCII is adequate only for standard American-English orthography. It does not even support complete British usage (words such as anæsthetic and œstrogen, and the £ sign), let alone the accents and diacritical marks required by European languages (élève, Würze, mañana), or the thousands of symbols required by Asian languages.

As a remedy, an international standard known as ISO-8859 has been introduced. This is an extension of ASCII, leaving all of the existing codes in place, but also assigning meaning to the 128 codes that ASCII (for historically valid technical reasons) did not use. This allows the more common accents and marks used in West European languages, but is still a source of great frustration for what it leaves out.

Another more significant extension is known as Unicode. Using more than a single eight-bit byte, Unicode can represent many thousands of characters and obscure symbols. Programmers are encouraged to use this *Wide character* representation for new development, but, due to compatibility issues, extra complexity in programming and processing, and simple human inertia, converted applications are still in a minority.

Business value proposition

The ASCII standard is universally known and almost universally used, and has been key technology for computing since its invention. The character set provides a common representation for interactions between computers and other equipment, and thus improves reliability of communications, reducing the need for error-prone conversions, and lowering the cost of technology ownership. ISO-8859 is the extended standard used by web browsers.

Summary of positive issues

ASCII is a very widely accepted and universally known standard, even surviving intact in the new ISO standards.

Summary of potentially negative issues

The standard is limited in its ability to incorporate a growing need for symbols and foreign characters. Even the Unicode standard leaves out many scientific symbols, thus requiring the use of nonstandard fonts for technical publications, which results in a significant risk of incorrect reproduction.

References

- ISO-14962–1997 (Geneva, International Standards Organization).
- ISO-8859–1 (Geneva, International Standards Organization).
- http://www.unicode.org.

Associated terminology: Dvorak/QWERTY keyboard.

Assembler

Foundation concepts: Compiler, Programming language

Definitions: Assembly code is the internal language of a CPU, rendered in human-readable form. An assembler is the system software that translates assembly code into the purely internal binary form used by the CPU.

Overview

Although C, C++, Java, and Basic are well-known and widely supported programming languages that programmers use to express algorithms in a form suitable for processing by a computer, they are not directly understood by computers at all. The instructions directly understood by computers, even the most modern ones, express only the exceptionally simple "baby steps" of a computation. They might command the addition of two numbers, or the copying of a single data value from one region of memory to another. It is these baby steps that are expressed in *Assembly language* programming.

In the early days of computing, before the development of *Compilers* (q.v.), programmers had to work at this micromanaging assembly language level. Computers can't understand human languages, so, in order to communicate, humans have to adopt the computer's own native language. This native language, when expressed in its pure internal, extremely user-*un*friendly binary form, is called *Machine code*, or sometimes *Machine language*. Assembly language is simply machine code with a few small concessions to human readers: each instruction is represented by a mnemonic word rather than in binary (e.g., "ADD" instead of "10000011" on a Pentium), and memory locations may be given names instead of numbers.

Assembly code made programming a very labor intensive and error-prone process. Additionally, each computer family has its own internal structure and assembly language, so the instructions used by a Pentium 3 to perform any given operation will be completely different from the instructions used by a PowerPC G4, or any other kind of processor, to do the same thing.

As higher-level languages such as *Algol*, *Cobol*, and *Fortran* were introduced, the need for assembly-language programming diminished significantly. An Algol programmer could write programs in a much more expressive language with automatic error-detection facilities, and the compiler software would translate that Algol program into assembly language automatically. Early compilers were not very efficient, so, for speed-critical parts of a program, programmers would still resort to assembly language in order to take complete control. High-level languages often do not provide support for machine-level operations, so operating systems implementers would also frequently use assembly language.

A modern, carefully designed compiler can almost always do a better job of producing fast, efficient assembly code than any programmer could in any reasonable amount of time. The notion that assembly language should be used for those parts of a program that require absolutely optimal performance is now almost always false, and most often counter-productive. It is generally only the creators of the most basic parts of an operating system who need to use assembly language at all, and then only in small amounts.

However, understanding of assembly language is still an important skill for any professional programmer. Even though it isn't directly used, assembly language reveals how a particular kind of computer works. Even a designer of rocket ships has to know how rivets work.

Business value proposition

The use of assembly-level programming has declined markedly in recent years as compiler techniques have improved. Even for special processors (e.g., those in cellular telephones), developers will primarily write their coded instructions at a high level and compile the solution down to small executable files which are burned into the ROM associated with the processor within the device. Assembly languages do, however, allow for closer manipulation of the processor and are always an option in extreme situations.

Summary of positive issues

A well-understood technology that has a mature and large literature associated with it. Today the primary users of assembly-level programming are operating-system and hardware developers.

Summary of potentially negative issues

Each processor type has its own individual instruction set, and learning to make effective use of an instruction set takes a long time even for an experienced assembly language programmer. Developing code in assembly language is extremely labor intensive and in most cases has no benefit. Debugging assembly-level programs is an exceptionally difficult process.

References

- Intel (1990). *i486 Microprocessor, Programmer's Reference Manual* (Emeryville, CA, Intel).
- G. Schneider, R. Davis, and T. Mertz (1992). *Computer Organization and Assembly Language Programming for the VAX* (Malabar, FL, Krieger).
- R. Paul (1999). *SPARC Architecture, Assembly Language Programming, and C* (Upper Saddle River, NJ, Prentice-Hall).

Associated terminology: Compiler, Programming language, C, Fortran.

Audio

Foundation concepts: Digital, Compression.

Overview

Sound is nothing more than rapid small changes in air pressure. Vibrating objects (such as larynxes and guitar strings) cause these changes, and nerves in the ear detect them. A microphone detects the pressure variations in a similar way to the ear, and converts them into corresponding changes in an electrical voltage. These voltage changes may be measured digitally, and represented as a series of numbers, which may of course be stored or processed by computer. The same sequence of numbers may be replayed, and, through a digital-to-analog converter, changed back into varying voltages and electric currents. When applied to a loudspeaker, the varying voltage creates a reproduction of the original sound.

The reproduced sound will never be a perfect reproduction of the original: digitization produces a *Quantization effect*,

which means that there is a smallest representable change and nothing smaller than it will be detected, and the varying voltages can be sampled only so many times per second. However, the precision of the human ear provides a very useful cap on requirements; there is no point in recording aspects of a sound that nobody is capable of hearing. If the voltage produced by a microphone is measured only 50 000 times per second, it will capture everything within the range of human hearing, and that is well within technological capabilities.

Once an audio signal is in digital form, it can easily be manipulated according to complex mathematical formulæ to produce effects that would otherwise require very expensive signal processing hardware. *Filtering* to remove noise, or to enhance a weak voice, or even to remove one voice from a group; pitch changes to enhance a voice or modify one aspect of a sound; and a wide variety of other transformations can all be applied with nothing more than a standard personal computer. Editing operations, in which parts of a recording are cut out and pasted together in a different order, removing any audible discontinuities, are also easily performed. *Scrambling*, or audio encryption, is also much simplified after digitization. Software for *Digital signal processing* is widely available for most computing platforms, and is relatively easy to implement for a software engineer with some knowledge of the mathematics of *Fourier transforms*.

Digitized audio signals stored *as is*, just as a sequence of numbers directly representing measurements of the sound pressure, are known as *WAV* or *Wave* files, although the correct technical term is *PCM* or *Pulse code modulation*. This is the format used on audio CDs, and is the most convenient for digital signal processing and playback. Wave files are rather large; a high-fidelity stereo recording requires about 10 MB per minute. This means that long recordings occupy a huge amount of disk space, and any application that requires audio signals to be transmitted over a network will consume a large amount of bandwidth. For this reason, there has been a lot of research into the compression of audio files, and a number of very effective methods are available.

There is sufficient redundancy in typical sound recordings that lossless compression (i.e., perfect compression: after decompression the restored data will be identical to the original) may sometimes halve the file size. Standard applications for producing "zip files" approach this compression ratio, and special-purpose lossless audio-compression software may fare slightly better. However, it is rarely necessary to reconstruct digital audio with perfect accuracy, since the human ear is not perfectly precise. Lossy compression methods allow the user to select the appropriate compromise between high compression ratio and high fidelity for any given situation. The well-known *MP3* (*MPEG Audio Layer 3*), *AAC* (*Advanced Audio Coding*), and *WMA* (*Windows Media Audio*) formats can reduce an audio recording to about 20% of its original size without great loss of quality, by progressively discarding the parts of the signal that are least likely to be detectable by human listeners. When the original audio contains no high-frequency components (e.g., spoken-word recordings) or when low quality is acceptable (e.g., cheap telephony), much greater compression ratios are possible.

Business value proposition

Since the advent of the internet, the use of sound on computers has radically departed from the past when computers were primarily silent devices. The primary use of audio is in entertainment and internet-related applications, and there are myriad applications, ranging from online instructional courses for employees, through

entertainment, advertisements, and assistance for the disabled members of society.

The basis of the technologies associated with audio on a computer is dependent upon the number of users in the interaction. One typical application is the *Webcast*, a unidirectional presentation made for example by company executives to analysts or shareholders. This requires that the recipients have an audio player on their computer that transforms the incoming signal into audible sounds; the webcast may or may not be accompanied by a video signal. Webcasts may be short in duration, such as those found in a shareholders' meeting, or they may be an endless streaming audio signal as broadcast by a radio station and "streamed" (distributed) via the internet. Bi-directional audio data transmissions, such as take place in *VoIP* telephone systems, require the sender to have a microphone–speaker set and programs to encode and decode the data.

Audio data (e.g., music) is also capable of being downloaded from the internet and stored on a portable computing device such as Apple Computer's popular iPods. These devices can download audio (the terminology has evolved such that a data stream is sometimes termed "Podcasting") in many formats.

The ability to provide audio data to computer users and mobile device users presents an opportunity to all owners of audio materials to provide their materials to a potentially paying audience, and substantially frees the owner from the traditional manufacturing and distribution costs associated with the delivery of audio materials. In some situations, there are clearly problems associated with protection of intellectual property and copyrighted materials; however, for those who want to be heard far and wide without restriction, e.g., politicians, it creates a low-cost means of distributing their materials to their audiences.

Summary of positive issues

Audio technologies are well developed and typically require only a computing device with a browser. The quality of audio transmitted on the internet is generally acceptable for most purposes, and high-fidelity audio is transmitted regularly as *Digital audio broadcasts* to digital radios in the UK and other countries. DAB combines MPEG and COFDM (Coded Orthogonal Frequency Division Multiplex) technologies to create a digital signal.

Summary of potentially negative issues

Internet-based audio requires compression to be used and has some loss in data quality. WAV and other formats have large data storage requirements. Intellectual property concerns surround the distribution of digital files.

Reference

- M. Bosi and R. Goldberg (2002). *Introduction to Digital Audio Coding and Standards*, Springer International Series in Engineering and Computer Science (Norwell, MA, Springer).

Associated terminology: Digital, Compression, Voice over IP.

Backup

Foundation concepts: Storage, Information lifecycle management, Power protection.
Definition: Secondary copies of data and applications kept to guard against catastrophic loss in the event of a system failure.
Not to be confused with: *BackUPS*, the trade name of a variety of uninterruptible power supply.

Overview

Hardware failures do occur. They can be made less common by using only the most reliable hardware components, employing all available *Power protection* technologies, and regularly monitoring system performance to catch problems before they occur, but nothing can prevent unexpected failure.

When a computer system suffers a hardware failure, it will sometimes leave data stored on disks undamaged, but some failures can destroy a disk drive, and some will result in damage to files: if an application is actively modifying a data file when the computer suddenly stops working, the file can be left in an intermediate state with just half of the change made (perhaps a balance transfer had credited one account but not yet debited the other), leaving the file's state invalid.

Sadly, amongst the most common forms of hardware failures are failures of the disk system itself. Damage may range anywhere from a slight degradation in performance to a complete loss of all data and the disk itself becoming unusable.

Additionally, software problems are very likely to cause loss of data. The presence of *Bugs* in applications is extremely widespread, and, if an application is "buggy," it could do anything from occasionally failing to update files completely to accidentally deleting everything stored on the disk. It must not be forgotten that *Viruses* are also a common problem; anything that a buggy application could do accidentally, a virus will happily do on purpose.

In short, any data kept on a computer is vulnerable to unpredictable total loss. Steps must be taken to protect all essential data. With a networked system, it is common practice to provide a centralized backup service: during non-peak times, data is copied over the network from individual computers to a second system, usually with enlarged storage capacity especially for this purpose. A network manager should be able to automate this process, so that it might occur in the middle of the night when the network is normally idle, and not require any human presence. Backup operations are often layered, so that perhaps every night only files that were created or modified the previous day are backed-up; every weekend, all files modified or created the previous week might then be re-backed-up to make a consolidated archive; perhaps every month the entire contents of every disk drive would be backed-up. This kind of arrangement reduces the amount of space required for backup copies, and also minimizes the search time required to restore data when necessary.

A central network-based backup system usually makes sense for any organization with more than one computer in use. It also requires even more stringent security than the other systems. Since this one computer holds a copy of all data from all other computers, it will be an obvious target for any break-in attempts. In addition, it must not be forgotten that the computer used for backups is just as delicate and fallible as any other computer. Losing all backed-up material due to a single hardware failure could be crippling for any organization. Clearly, the archive files created by backup procedures can not be left long-term on a regular disk storage system.

Simple computer or software failures are not the only causes of data loss. Fires, floods, earthquakes, tornados, riots, and terrorist attacks can all leave an entire installation completely destroyed. Any organization that is to survive such a disaster

must store its backup archives in a location that is both secure and distant. The same principle applies to non-networked computers: the only safe solution is to make copies of the backup archives on some removable storage medium, which should then be removed to another location.

Choosing a suitable removable storage medium is a serious problem. The simplest and cheapest solution is to use writable CDs (known as CD-R, CD-RW, and CD-RAM). High-speed drives typically cost less than $100, and, when bought in quantity, the blank disks cost only a few tens of cents. However, the capacity of a CD is only 650 MB. That can be stretched a little by the careful use of data *Compression*, but is only really adequate for the frequent *Incremental* backups that save only those files that have changed recently. It would take 250 CDs to perform a full backup on a typical well-used disk drive.

Another alternative is the writable DVD (known as DVD−R, DVD−RW, DVD+R, DVD+RW, DVD−RAM, DVD−DL, and DVD+DL). These are in effect simply CDs with higher capacities. They are a little slower, and rather more expensive, but from the user's perspective fundamentally the same. The capacity of a DVD is 4.7 GB, or, for the much more expensive dual-layer (−DL and +DL) versions, 9.4 GB: seven (or fourteen) times the capacity of a CD. So DVDs can hold more incremental backup material, but would still not be suitable for backing up entire disks.

Magnetic tapes in various forms provide another alternative. Until quite recently, magnetic tapes were seen as the perfect solution to the backup problem, since a single tape could store the contents of a few complete disk drives. Unfortunately, disk technology has improved far more rapidly than tape technology, and the ratio has been reversed. Currently, tape cartridge drives with capacities between 1 and 400 GB and greatly varying prices are available, and auto-loading units that can hold ten tapes at a time and automatically eject one and load another when necessary extend the capacity to genuinely useful sizes. Providing adequate backups of data is a technological problem that can not be ignored or solved by default.

There are also legal considerations. Backup archives containing copies of confidential data must be protected at least as carefully as the original. Old tapes that have reached the end of their working lifetime or that are no longer needed can not simply be thrown away; they must be properly destroyed and made unreadable, or the data must be completely overwritten first. In nearly all circumstances it is permissible to make a copy for backup purposes of purchased copyrighted software (commercial applications and operating systems, for example) because CDs, even those used by reputable companies to distribute expensive software, do sometimes fail or break. However, care must be taken that illicit installations are not performed from the backups, and, if the original software is sold or given to another party, the backups of it must be destroyed or erased.

Business value proposition

The correct and successful backup of data is vital to any organization and thus the ultimate responsibility to ensure compliance falls on the CIO. The task itself may technically fall under the responsibilities of the network manager with input from the chief security officer. Typically the backing up of data is considered through a process known as *Information lifecycle management* (*ILM*) in which the data is guided all the way from creation to disposal through a series of processes and technologies that support the value and nature of the data at each point in time. For example, transactional data may be archived incrementally (only the changes are archived) to a backup tape server until the data is extracted from the transactional data, transformed and then

loaded into a data warehouse, at which point the data may be backed up incrementally onto tapes and these removed to another location for safe keeping. The mechanisms and forms of the data backup will also depend upon the scale of data being backed up. For a small company the capacity requirements may be such that a CD or DVD provides sufficient storage capability for a weekly or monthly backup. For those companies which require a complete 100%-redundant systems capability there are specialist computers available from various manufacturers, which contain two or more of everything with identical data processing and storage occurring in each of the two systems; thus, should one CPU or disk drive fail, the system can continue to run as normal in the other system while the problem is corrected. RAID (redundant array of inexpensive disks) techniques provide similar concurrent backups for disks only.

The legal issues facing many corporations, particularly those operating in the United States, dictate the need for strong backup policies, acts such as Sarbanes–Oxley and HIPAA require data not only to be stored correctly but in some instances also to be disposed of correctly. Again these issues need to be resolved by the organization in the form of processes and policies pertaining to data backup. Clearly legal issues are important, but the possibility of loss of intellectual property or even government secrets dictates that data must be stored in the most appropriate way (e.g., using encryption), and disposed of securely (e.g., by passing a large electromagnetic pulse through the system or at the very least writing to the data storage medium several times until the whole data-storage area has been covered and all prior data made unreadable and incapable of being extracted through any special forensic programs).

There are many third-party companies, vendors, and consultants to support the backup processes surrounding the information lifecycle.

Summary of positive issues

The backup requirements of organizations can be related to well-known and established models such as the information lifecycle. Technical issues surrounding the establishment of backup servers, use of tandem systems, off-site disaster recovery, regulatory compliance, and security are well understood, documented, and supported.

Summary of potentially negative issues

All data-storage media are fallible. Data storage and backup systems require the expenditure of resources and capital. The creation of local data that is not supported by a network server may not be subject to or adhere to policies that enable data to be recovered through backup procedures. Some of the technologies such as CD and DVD are limited in capacity and therefore unsuitable for large-scale backups at data-rich organizations. Some data storage media have a wide range of sub-categories and require care in the selection of devices. Failure to adhere to regulations pertaining to data backup and storage can lead to prosecution.

Reference

- D. Cougias, E. L. Heiberger, and K. Koop (eds.) (2003). *The Backup Book: Disaster Recovery from Desktop to Data Center*, 3rd edn. (Lecanto, FL, Schaser-Vartan Books).

Associated terminology: Bug, RAID, Information lifecycle management.

Bandwidth

Foundation concept: Bit.
Definition: The maximum rate of information transfer.

Overview

The quantity of information is measured in bits. One bit is the smallest amount

of information possible; it represents the answer to a yes/no question. All data can be represented as a sequence of answers to yes/no questions.

For example, if someone is having a baby, and you want to be told whether it is a boy or a girl, someone could transmit to you one of the phrases "*it's a boy*" or "*it's a girl*," or you could decide in advance that you want the answer to the question "*is it a boy?*," and then they could transmit to you exactly the same information with either "*yes*" or "*no*." In computing of course, yes is traditionally represented by the digit 1, and no by 0. If the question is decided in advance, one single digit, one bit, gives all the information.

On the other hand, the question may be more complex. Perhaps someone is buying a new car, and you want to know what color it is. You already know that it has to be red, blue, green, or yellow, and want to be told which. Again, you could decide in advance that you want the answers to two questions: first "is it red or blue?," then "is it blue or yellow?" If the car is red the answers are "yes, no"; if it is blue the answers are "yes, yes"; if it is green the answers are "no, no"; if it is yellow, the answers are "no, yes." If the language of communication (the questions to be answered) is decided in advance, an intrinsically non-yes/no question, "what color is it?," can be answered with two yes/no answers.

Every possible piece of information can be encoded as a sequence of yes/no answers. In simple cases, when there are only two possible answers, only a single bit is required. When there are only four possible answers, two bits are required. In most human communications, there is a very large number of possibilities, but still everything can be covered by a sequence of yes/no transmissions. A predetermined eight-bit encoding based on the ASCII table can be used for sending arbitrary text. For example, the letters A, B, C, D, E are represented by the sequences 01000001, 01000010, 01000011, 01000100, 01000101, respectively, so the word BAD is represented by the 24-bit sequence 010000100010000101000100. In this way, the entire bible can be represented by a sequence of approximately 35 000 000 1s and 0s.

The rate at which data is transmitted is measured in bits per second, and is limited for any physical medium. The maximum rate for any medium is called its *Bandwidth*. The term comes from radio. A radio station is allocated not just a particular frequency, but a whole continuous band of frequencies: an AM station nominally transmitting on 200 kHz would in reality be transmitting on all frequencies in a band between 195 kHz and 205 kHz. This gives it a bandwidth of 10 kHz (i.e., 10 000 cycles per second), which determines the maximum rate at which it could transmit data. Wider bandwidths give larger data rates.

Frequently used derived units for bandwidth are: *Baud*, equivalent to one bit per second including overhead, not just useful data; *Mbps*, *megabits per second*, equivalent to 1 000 000 bits per second; *Gbps*, *gigabits per second*, equivalent to 1 000 000 000 bits per second; *Kbps*, *kilobits per second*, equivalent to 1000 bits per second (this is not generally the same "K" as is used to measure memory size). Confusingly, the "bps" abbreviation is occasionally used to mean bytes per second instead of bits per second, but this is not the normal usage. The phrase "per second" is often omitted: a "gigabit network" is one with a bandwidth of 1 000 000 000 bits per second.

If a communications medium has a particular bandwidth, it is physically impossible to transmit more data per second. Using a 100-megabit ethernet connection, it is physically impossible for more than 100 million bits to be transmitted per second.

Data may be compressed, which simply means removing unnecessary or redundant details, transmitting only what really matters, and adding the redundant parts back

in after reception. Returning to the original example, the string "YES" is a 24-bit message, but if it is known in advance that the only possible strings are "YES" and "NO," the transmission can be reduced to one bit, 1 for "YES" or 0 for "NO." Just one bit is transmitted, but the receiver converts that one bit back to "YES" or "NO" before displaying the result. There was only one bit of real information in the original 24-bit message, so a compression ratio of 24 to 1 was possible.

Bandwidth limitation is one of the fundamental requirements of information theory; no reputable authority disputes it. If an invention claims to give you a transmission rate of 100 MBits per second on a 10 MBit line, it *must* be doing so by compressing the data, removing redundant information. If the data you are transmitting does not have a 90% redundancy rate, the invention can not work.

Business value proposition

Organizations need to determine both their internal and their external bandwidth requirements. The determination of internal bandwidth requirements is based upon the amount of data transmitted internally within the organization across the LAN. External bandwidth requirements are determined by considering the data requirements of the company in relation to any customers, vendors, or other entities with which it interacts electronically.

In reality, determination of bandwidth requirements for companies can be a complex task. Typically network managers and CIOs are forced to trade off performance against cost. The total cost of ownership will be calculated inclusive of issues such as the future bandwidth requirements for a growing organization, the scalability of the network connections, the type of data traffic, and the growth rates associated with the different types of data traffic placed upon the networks.

Organizations must also fully determine the drivers of the external demands for bandwidth requirements. A business-to-business (B2B) e-commerce marketplace for example, wholly dependent upon its online customers, would typically be concerned with metrics such as the end-to-end response time experienced by its users when accessing and interacting with the marketplace (end-to-end response time refers to the time it takes to send a request to a site and then receive a response back). To ensure customer satisfaction the B2B company world need to invest in a network that has a bandwidth provision in excess of that needed to accommodate the demands of a peak user population interacting with the marketplace.

Network facilities are often purchased from third-party service providers such as ISPs or web-hosting services. Care needs to be taken when choosing a service provider to ensure that the bandwidths and services contracted are actually provided. For example, a web-hosting service may offer a "bandwidth" of 10 MBytes per day, but that does not mean that the access speed is high, it simply means that your customers may download up to 10 MBytes from that site per day without incurring extra costs for you. Hence contracts need careful management and examination.

Summary of positive issues

The cost of bandwidth and networking equipment is continuing to decrease in real terms. As the availability of bandwidth continues to grow, the number of devices that will be capable of connecting over a network will continue to grow.

Summary of potentially negative issues

Total cost of ownership needs to be completely understood in order to determine the system requirements correctly, since underestimation can have an extremely negative consequence for the organization.

Reference

- C. Lu (1998). *The Race for Bandwidth: Understanding Data Transmission* (Redmond, WA, Microsoft Press).

Associated terminology: Compression, Bit, Binary, Digital.

Bar code

Definition: A uniform product code (UPC) that uniquely identifies an object, encoded as a sequence of visible stripes or bars that are easily read by machine.

Overview

The origins of the bar code stretch back to 1949 when Bernard and Norman Silver filed a patent application titled "Classifying Apparatus and Method." They devised a disk that had concentric circles on it, which could be read under ultraviolet light. The modern "bar code" formally known as the Uniform Product Code (UPC) was devised in 1973 by George J. Laurer, a researcher at IBM.

Each digit 0–9 corresponds to a unique pattern of narrow and wide bands; an entire bar code simply represents a large number. The UPC system assigns large sequences of numbers to different manufacturers, and within their own sequences each manufacturer assigns unique numbers to each of their products. This ensures that no two products will ever have the same UPC, and a simple and cheap method exists for scanning bar codes and retrieving information on the product it is attached to.

Two forms of bar code exist; the United States and Canada use the UPC, while the European Article Numbering (EAN) format is used elsewhere.

Bar codes can be used to represent any numeric data, and therefore anything that has a digital representation. They have been used experimentally to publish TV-guide information readable by VCRs with special scanners, and even to publish small pieces of software. Applications such as these seem attractive because bar-code scanners may be constructed very cheaply, especially when compared with the costs of a full OCR-quality flatbed scanner. Unfortunately, only small amounts of data can be encoded as bar codes before they become prohibitively oversized.

Business value proposition

Bar codes representing UPC and EAN codes require a laser or other optical scanner. When EAN devised their data format they expanded the scope of the codes and made UPC a subset of EAN; this enables EAN scanners to read both formats but not vice versa. Bar codes have been very useful in the automation of manufacturing, logistics, and, of course, retail operations, where the ubiquitous point-of-sale (POS) scanner speeds customers through the checkout at stores.

Summary of positive issues

The technology associated with bar-code format is well established and understood, linking into inventory-management systems and ERP systems.

Summary of potentially negative issues

UPC codes have scalability constraints and are used only in the United States and Canada. The growth in usage of radio-frequency identity (RFID) tags is expected to first complement and then possibly replace the bar-code labeling mechanism.

References

- R. Palmer (2001). *The Bar Code Book: Comprehensive Guide to Reading, Printing, Specifying, and Applying Bar Code and Other Machine-Readable Symbols* (Petersborough, NH, Helmers).
- S. Pearce and R. Bushnell (1997). *The Bar Code Implementation Guide: Using Bar*

Codes in Distribution (Philadelphia, PA, Tower Hill Press).

Associated terminology: RFID, Optical character recognition.

Batch processing

Foundation concept: Operating system.
Definition: The automatic execution of a series of predetermined commands by a computer system to complete a job without human intervention.

Overview

The term "batch processing" originated in the days when all the inputs to computers were in the form of *Punched cards* (also known as *Hollerith cards*, cards of just over $7'' \times 3''$ with holes punched in them to represent typed characters, each card representing a line of input), paper tape, or magnetic tape, often without any interactive input at all. For each *Run* or *Job*, a *Batch* of cards was prepared, commanding the computer to run various applications in a particular sequence, directing input and output to various places, and providing contingency instructions for handling error conditions. Batch processing in this form allows very efficient use of computer resources, since the system is never left idle waiting for human activity.

The technology has now changed significantly, and for most computer systems the time wasted when the computer is awaiting user commands is not considered a valuable resource. However, there are two cases in which the principles of batch processing, if not its original details, remain in use. When an organization has invested significant capital in a large mainframe or supercomputer for special-purpose processing, its time may still be in great demand, and having it follow a predetermined sequence of commands, rather than operate under direct human control, is a sensible decision. Secondly, when a complex sequence of steps is required, and a human operator is likely to make mistakes, or when a sequence of steps is to be performed repeatedly or at inconvenient times (such as night-time backup procedures), automated control is valuable. In both of these cases, the commands are not punched onto cards, but typed into command files, which the operating system is directed to obey at a specified time. These files are known as *Batch files* (*BAT* files under Windows), *Command files*, or *Scripts*.

Business value proposition

Batch processing is still a viable option when attempting to optimize the processing of certain data sets. Not all data needs to be processed in real time and the use of batch methods allows the systems administrator (who performs a similar set of tasks and functions to the computer operator of the 1970s) to optimize the machine's usage, such as running the batch process on the system at night when other usage is low. Alternatively, the systems administrator in conjunction with the CIO may determine that they wish to outsource the processing of certain batch processes if this makes economic or operational sense (this again is the same as when companies sent tapes or batches of cards to computer bureaus in the 1960s and 1970s). Finally, batch processing still occurs in modern ERP systems, when they perform tasks on accumulated data in a single intensive session, such as creating a consolidation of accounts at the end of a financial reporting period.

Summary of positive issues

Batch processing allows optimization of the computing environment for certain functions, and may provide an organization with a simplified option for the outsourcing of a process.

Summary of potentially negative issues

Batch processing operations still need careful monitoring to ensure that they do not

tie up valuable resources during periods of peak demand on the computing environment during their run, and to make sure that the job is completed successfully.

Associated terminology: Legacy system, Transaction processing.

BCS (British Computer Society)

Definition: The British Computer Society (www.bcs.org) was founded in 1957 and acts as an industry body to promote professional services within the IT sector and provide resources to encourage greater public awareness of the social and economic dimensions of information technology.

Overview

The BCS is a professional society based in the UK that aims to provide services, advice, and a career development path to its members and the IT sector as a whole.

The BCS sees its mission as one of raising awareness of IT as a profession and ensuring that technology development is undertaken with the highest levels of integrity and competence. The society has a well-developed pathway for professional development consisting of membership levels that are based upon academic qualifications and experience. Standard grades include Companion (ComBCS) for students and Associate (AMBCS) for affiliates, with Member (MBCS) and Fellow (FBCS) being considered as professional grades. The awarding of Chartered Professional Status is also available for senior members through the Chartered Engineer (C.Eng.), Chartered Scientist (C.Sci.) and European Engineer (Eur. Ing.) designations which confer professional status levels recognized in the UK and the EU.

The BCS also accredits university courses in computer science and IT, which can lead to exemptions from BCS exams for those candidates aiming to gain professional status.

Business value proposition

The BCS provides a range of services for the IT profession that are of the highest quality, including professional development courses, scholarly journals, conferences, and sponsorship of specialty groups. The society has a large number of branches throughout the UK and the world that provide academic and practitioner seminars and lectures to members and facilitate networking opportunities for members.

The society also facilities the "Elite Group" meetings, a forum for UK IT directors and senior executives to exchange views and discuss their technology-related experiences.

The membership levels of the society can be used as a mechanism for judging professional merit of individuals during corporate recruitment, the levels reflecting the academic as well as practical experience of the holder, although many highly competent professionals choose not to pursue membership. Members are held accountable to a professional code of conduct and the BCS acts as arbiter of that code. The BCS also maintains a Professional Advice Register of consultants, security specialists, and expert witnesses.

Summary of positive issues

The society has a wide reach through its branches and its professional development courses. Its membership levels are well embedded in the IT profession within the UK.

Summary of potentially negative issues

There is no formal requirement to join a "professional" organization such as the BCS, and membership is still optional for computing and IT professionals, unlike for the older professions of medicine (American Medical Association, General Medical Council) and law (the American Bar Association). The Pan-European qualifications and Chartered Scientist titles are not widely understood beyond the EU.

References

- http://www.bcs.org/BCS/.
- The British Computer Society, 1 Sanford Street, Swindon, Wiltshire SN1 1HJ, UK.

Associated terminology: ACM, AIS, IEEE, W3C.

Benchmark

Definition: A standardized set of tests, performed uniformly on a range of different systems, to produce an objective comparison of performance.

Overview

When selecting a new computer system for any performance-critical task, it is essential to have some clear, consistent, and objective measurement of the computing power of each of the candidate systems. Contrary to common belief, the advertised clock speed of the CPU (figures such as 2.88 GHz) does not provide a reliable measurement of performance. One kind of processor may be able to do more in a single clock cycle than another, and a poorly designed motherboard or slow memory can ruin a nominally fast processor.

A *Benchmark* is a purpose-built application, or a predetermined sequence of operations to be performed by an existing application, designed to get an accurate picture of the speed of the computer as a whole while it is performing exactly the kinds of tasks that it will be used for. The results of benchmark tests are often called *Performance metrics*. Trade magazines frequently perform their favored benchmarks on new systems when they are released for review, and may provide a very cheap source of such information.

Some of the most popular benchmarks are the "Business Winstone," which measures the speed of the most popular Windows applications in realistic situations; "Winbench 99," which measures the performance of the system's major components; "Netbench," which measures the performance of a file server; and "Dhrystone," which measures the speed at which simple arithmetic operations are performed. There is some popularity in expressing a CPU's speed in terms of "Flops," the number of "floating-point operations" (i.e. more complex arithmetical operations) it can perform in a second (the MegaFlop is one million Flops), but this measurement is primarily of interest for scientific applications, and is of limited value in the business and data processing environment.

Business value proposition

Organizations need to understand the realities associated with the benchmarking of any systems that they contemplate acquiring, and which will be specified in the purchase contract. Performance benchmarks supplied by vendors need to be examined with respect to the true operational context of the corporate IT architecture as a whole when the system is interacting with the live network, other applications, etc. Benchmark "stress" testing is especially useful since failure usually occurs when the systems are working at their operational limits.

Summary of positive issues

Software tools and consultant support are available to facilitate benchmarking.

Summary of potentially negative issues

The benchmarking process can be resource-intensive and as complex as the scale of the system being tested. Failure to undertake correct benchmark testing prior to acquisition can lead to catastrophic situations, e.g., when a critical ERP module can not handle the volume of data requests and database updates the whole ERP system may fail, possibly even corrupting the database.

Reference

- D. Lilja (2000). *Measuring Computer Performance: A Practitioner's Guide* (Cambridge, Cambridge University Press).

Associated terminology: Software metrics, Efficiency.

Binary

Foundation concepts: Bit, Digital.
Definition: A data representation in base two.

Overview

People do not usually distinguish between the abstract idea of a number and its representation. The connection between the abstract idea of the number forty-two, as the total number of spots on a pair of dice, or the number of days between November 5 and December 17, and the numeric representation "42" is logically quite tenuous. If people didn't have ten fingers, our number system would be quite different; if we didn't have eyes, we would not use inked shapes as the representation at all.

It is quite possible to build a digital computer based on our familiar decimal number system (in the 1950s this approach was not uncommon); it is easy to construct electronic circuits that can accurately distinguish among ten different voltage levels, and that is all that is required to represent the digits. It is also possible to construct circuits that perform additions and multiplications in this form. However, it is much, much easier if only two different levels need to be distinguished. It is very easy to tell the difference between "on" and "off," orders of magnitude easier than distinguishing among ten different levels of activity.

If there were only two digits instead of the familiar ten, all of arithmetic would be much easier. How long would it take to learn the multiplication table? "One times one is one," and that's it! Electronic circuitry is well over a hundred times simpler, and therefore cheaper, if it has to encapsulate only the four facts "$0 \times 0 = 0$, $0 \times 1 = 0$, $1 \times 0 = 0$, $1 \times 1 = 1$," instead of the two hundred more complex facts that school-children have to learn (e.g., "$7 \times 8 = 6$ carry 5").

So all modern computers use the binary system, based on twos, instead of our familiar decimal system based on tens. In decimal each digit is worth ten times as much as the digit to its right, so the four ones in 1111 are worth, respectively, one thousand, one hundred, ten, and one; the whole number is one thousand one hundred and eleven. In binary each number is worth only two times as much as the digit to its right, so the four ones in 1111 are worth, respectively, eight, four, two, and one; the whole number is fifteen.

Every operation is easier in binary than in decimal, even for people. The one disadvantage is that numbers need to be longer: with ten different digits to choose from, a seven-digit phone number gives ten million different combinations; with only two different digits, it would give only 128 different combinations. In order to serve ten million people, binary phone numbers would need to be 24 digits long. People have severe problems trying to remember a sequence of 24 ones and zeros; computers are not similarly encumbered by their mental architecture.

Business value proposition

The binary system is the basis of all modern (post-1950s) computing and all devices and systems are based upon this, the only exception being analog input devices that pass their output through a D/A (digital/analog) converter. This approach to technology allows compatibility, reliability, and data interchange between devices and relieves organizations of the need to check for compatibility at the most basic level of data representation.

Summary of positive issues

The binary system is universally accepted as the basis of computing and computing devices.

Summary of potentially negative issues

The binary system is easy for computers to understand but very difficult for humans to interpret and process on any scale other than trivial examples. This has caused computer scientists to develop higher-level programming languages and data encoding mechanisms.

Biometrics

Definition: "Life measurements": measurements of the physical aspects of a human subject's appearance, used to enable automatic recognition and verification of identity.

Overview

Recognition of individual people is a very complex operation, made more difficult by the fact that people do it so naturally and unconsciously that we can not analyze our own methods. Explaining how you recognize a familiar face is like trying to explain how you make your heart beat, and provides no insights that are of use to a software designer.

At first sight, the problem seems nonexistent. People normally think they know exactly how they recognize others. We remember hair and eye color, height, moles, and facial expressions, and it is tempting to assume that we recognize a person by matching them with a set of known characteristics. Unfortunately, that isn't the way it works. We still recognize our friends when they wear hats and close their eyes. More tellingly, we are not fooled by disguises when an interloper tries to look like a familiar person.

Establishing identity is an important part of all computing enterprises. Automated recognition may be simply a convenience, so that a user does not have to waste time with self-identification or usernames and passwords. It may be a matter of security, where critical information can not be released to somebody simply because they managed to find out what the password is. When strong encryption methods are used, it is impossible for anyone to remember their own encryption keys; they are numbers with many hundreds, sometimes thousands, of digits. Strong encryption and decryption keys *have* to be electronically recorded, and that makes the problem of protecting that recording, so that only the true owner may access their own encryption keys, absolutely critical.

Simply comparing a captured digital image with a pre-recorded one is not a viable solution. The temporary changes that happen to a person's appearance as a result of normal life, such as hair cuts, strong wind, rain, allergic reactions, wearing contact lenses, new lipstick, etc., are much greater than the permanent differences in appearance between two distinct individuals. An image-based system that is capable of recognizing the right person under varying circumstances must have so much latitude that it will also falsely recognize many other people.

Biometrics is the idea of making quantitative measurements of a person's physical attributes for use in future recognition or identity verification. It is usually not the actual measurements that are used, but the relationships between them. For example, the distance between a person's eyes may be useless, as it varies with range and angle of viewing, but the ratio of the distance between their eyes to the distance between the earlobes is almost invariable for any given person, and very hard to fake. Fingerprint recognition is another aspect of biometrics: exact matching of pre-recorded fingerprints will be thwarted by small scratches and ink blots, but the detection of major features

such as whorls and ridges, and their relative sizes and positions, can be quite successful.

Biometric systems are still very much in their infancy. There is no established accepted set of biometric parameters that may be used for reliable recognition, and the details of many existing systems are jealously guarded commercial secrets. A low-cost biometric system is unlikely to be of any use except as a gimmick or symbolic deterrent. Even high-cost ones need to be thoroughly tested before any investment is made.

Business value proposition

Biometrics systems can be purchased and used to secure a variety of products and processes. These include fingerprint systems for doors, computers, and any device that can be loaded with software and connected to a USB fingerprint reader. The use of facial recognition biometric systems provides a convenient, cost-effective mechanism for the provision of a high level of security. Other biometric systems such as iris recognition, hand geometry, signatures, and voice prints are developing in their levels of reliability and acceptance.

Summary of positive issues

Biometric technologies have matured and "smart cards" with photographic biometric and cryptographic technologies are now required by agencies such as the US government's federal agencies for personnel identification (Homeland Security Presidential Directive 12). Technologies such as fingerprint readers have become accessible and low-cost solutions are available. The biometrics industry is supported by a large consulting base and academic literature.

Summary of potentially negative issues

Biometric security solutions can require a significant investment in ongoing administration, monitoring, and support. Biometric technologies and solutions are continuing to develop and are not infallible.

Reference

J. Woodward, M. Orlands, and P. Higgins (2002). *Identity Assurance in the Information Age: Biometrics* (New York, McGraw-Hill).

Associated terminology: Encryption, Security.

Bit

Foundation concepts: Binary, Digital
Definition: A single binary digit, zero or one.

Overview

The smallest piece of information that can exist or be communicated is the bit; a single symbol 0 or 1.

The meaning of the symbol is completely context-dependent: it could represent "no/yes," "off/on," "the number I am thinking of is less than/not less than 1024," or the resolution of any other dilemma. Usually a single bit is just part of a larger communication, but any information can be represented as a series of 0s and 1s (see *ASCII*, *Binary*). Information quantity is always measured in bits and the usefulness or carrying capacity of any channel of information is known as its *Bandwidth* and is measured in bits per second. This is true whether the channel is a digital fiber-optic cable, a high-definition color-television broadcast, a person tapping on a morse-code key, or the human voice.

Business value proposition

Bits provide the common basis of information measurement and representation for all digital computing, networks, and computing devices.

Summary of potentially negative issues

Computing when considered at the individual bit level can be extremely data intense

and has necessitated the development of higher-level tools, representations, and languages in order to manipulate the data efficiently and effectively.

Associated terminology: ASCII, Bandwidth, Compression.

Bluetooth

Foundation concept: Wireless network.
Definition: Bluetooth is a specification (protocol) for wireless data transmission between devices that support that protocol.

Overview

Bluetooth is a protocol for wireless communication between devices of various kinds, and has been designed to be economic in terms of processing power requirements. The protocol has its origins in a Special Interest Group (SIG) created in 1988 that took its name from the tenth-century Danish king Harald Blatand (Harold Bluetooth in English). The SIG was focused upon the development of an open standard for wireless communication between devices operating within a short range of each other and having a low power consumption requirement.

The system works by wirelessly connecting two or more devices in what is known as a *Piconet*, using a base band frequency of 2.4 GHz. The system has been designed to work in noisy radio-frequency environments, where there is already a lot of radio-frequency energy being transmitted and less robust systems may be unable to work through the interference. It achieves this by employing a technique referred to as *Frequency hopping* whereby the master device communicates to the other devices (known as *slaves*) in its piconet first at one frequency and then at another, hopping through the frequencies during the communication. If some of the frequencies are too noisy, and the communication fails, then it will still be successful on others.

Bluetooth occupies layers 1 and 2 of the OSI seven-layer model, and thus may be used in place of network cards and cables in a *local area network*. Normal TCP/IP network traffic may be carried over Bluetooth devices. In this capacity, Bluetooth is a popular means of adding mobile internet connectivity to a portable computer by interfacing it with a suitably enabled cellular telephone.

Business value proposition

Bluetooth is an open standard administered by the Bluetooth SIG (https://www.bluetooth.org/) that has been adopted by manufacturers and employed in a wide variety of devices, including wireless headsets, wireless-enabled telephones, computer mice and keyboards, printers, USB adapters, "Bluetooth access points" that allow connection to a wireless network, and wireless office equipment such as electronic whiteboards, as well as Bluetooth-enabled gaming consoles, medical equipment devices, and GPS devices.

A primary adopter of the technology has been the automobile industry, to facilitate device-to-device communication, focusing upon hands-free cell-phone operation through the vehicle's voice-recognition system.

Summary of positive issues

Bluetooth is an open standard that facilitates communication between devices at a short distance using a frequency-hopping technique that enables it to work in noisy radio-frequency environments such as an office or home. Bluetooth has three levels of security and authentication systems built into it. Multiple piconets can be combined into one *Scatternet*.

Summary of potentially negative issues

The Bluetooth system works at a close proximity (up to 100 m depending upon the power class of the system) and has a

limit on the number of devices that can communicate on one piconet (one master and seven slaves). The bandwidth of the specification is low (1 Mbps). There is a practical limit to the number of piconets and scatternets that can be in very close proximity due to interference from overlapping signals.

References

- http://www.bluetooth.com
- R. Morrow (2002). *Bluetooth: Operation and Use* (New York, McGraw-Hill Professional).

Associated terminology: OSI seven layer-model, TCP/IP.

The Bluetooth brand is owned by Telefonaktiebolaget LM Ericsson.

Broadband

Foundation concept: T-Carrier.

Definition: Broadband is a general term that refers to a high-speed data-transmission link.

Overview

The term *broadband* does not imply any specific minimum bandwidth for data transmission over a physical cable. Broadband is a term generally used to describe any high-speed link for the transmission of data or voice in a residential setting, that has a bandwidth higher than a dial-up internet connection.

Dial-up internet connections are based on the use of analog modems to connect a computer to an internet service provider (ISP) over the normal cables owned by the telephone-service provider. Dial-up services have been available to the general public since before the internet was deregulated in 1995 at a variety of bandwidths depending upon the technology used. The base bandwidth for dial up is 56 Kbps (Kbps = thousand bits per second).

Broadband technologies are based upon the transfer of data as a digital signal. Several technologies have evolved to provide this service. One such technology was termed an *Integrated Services Digital Network* (ISDN) and is an ITU-T standard. Two levels of ISDN have been defined: the basic rate interface (BRI) allows for two channels of 64 Kbps or one of 128 Kbps; the second level of ISDN is known as a primary rate interface (PRI) and can carry 30 channels of data at 64 Kbps per channel, providing a total bandwidth of 1.92 Mbps. These channels known as "bearer" or "B" channels can be leased wholly or in part.

A second broadband technology known as the T system (T−/+1, T−2, T−3, etc.) is discussed in the *T-Carrier* article.

A third broadband technology is known as the digital subscriber line (DSL), a mechanism for transmitting digital data over the standard copper telephone wire found in most residential areas. Since DSL works over a telephone line, it has the advantage of being a dedicated (not shared) line. There are several varieties of DSL, based on the technology used and the distance of the end user from the service provider. The speed of DSL broadband connections is often asymmetrical (ADSL), meaning that transmissions to the user (downloads) have a different bandwidth from transmissions from the user (uploads). Typically, residential users download much more than they upload, and the asymmetry reflects this. Typical downstream rates are speeds of up to 1.5 Mbps while upstream rates of 640 Kbps or less are common. At the other end of the DSL bandwidth spectrum is very-high DSL (VDSL), which uses fiber-optic cables to the end user, providing downstream bandwidths of 55 Mbps for lines of up to 300 m in length and 13 Mbps over lengths beyond 1500 m, with upstream bandwidths of 1.6–2.3 Mbps.

A broadband connection can typically also be obtained through residential cable television service providers. This requires the use of a cable modem rather than a telephone modem. While bandwidth typically

ranges from 0.5 to 3.0 Mbps, cable modems use lines that may be shared by a very large number of other customers, and, as with all networking systems, the available share of bandwidth can be drastically reduced at times of peak demand.

Business value proposition

The utilization of broadband provides its user with a higher-bandwidth connection to the internet than traditional dial-up technologies provide.

Summary of positive issues

Broadband is available at a variety of bandwidths to home and commercial users.

Summary of potentially negative issues

Broadband speeds need to be checked by programs that specifically examine upstream and downstream data rates. Cable access to broadband is limited in commercial office buildings because these facilities tend not to have been wired for television.

References

- http://www.itu.int/.
- http://www.dslforum.org/.

Associated terminology: Bandwidth, Network.

Bug, debugging

Foundation concept: Software development lifecycle.
Definition:

Bug (1): An error in some software.
Bug (2): A remote surveillance device.
Bug (3): A movable pointer on an instrument dial, avoiding memorization of settings.
Debugging (1): Attempting to detect and remove errors in software.
Debugging (2): Attempting to detect and remove surveillance devices.

Overview

1: Software never goes wrong. If software does not behave as it should, then it was wrong right from the beginning. Either it was installed incorrectly, or it was wrong when it left the factory. Apart from the possibility of software being corrupted by a computer virus or accidentally by some other piece of incorrect software, software does not change in use; it remains immutable. It can not *go* wrong. These distinctions are, of course, of little help. It is well known that software very frequently *is* wrong. Knowing where to place the blame does not solve the problem.

It is almost true to say that all software is wrong. Every large piece of software seems to have something wrong with it, usually many things, many of them significant. This is an entirely human problem. It is not part of the nature of software to have bugs, it is the nature of software development methods to produce them.

Human beings are exceptionally fallible creatures. We may be capable of all sorts of clever things, but our memories are very idiosyncratic. Developing a large piece of software requires programmers to remember incredible amounts of detail about hw each component is to be used, and how they are intended to interact. If more than one programmer is involved, the vicissitudes of memory are abetted by the ambiguities of communication: how do you know that what you understood someone to say is exactly the meaning intended?

Over the years, numerous software development methodologies have been developed, with the intent of putting some order into the process of programming, providing checks and balances and requirements for documentation (see *Software development lifecycle*). All of these methods still rely on humans doing what they are supposed to do. No matter how well set out the rules are, there is always the possibility that a

human being will actually be human and make a mistake, failing to follow the procedure and introducing the very errors that those procedures are supposedly guaranteed to prevent.

There is one approach that provides a theoretical chance of success: *Formal methods* together with *Computer-aided design* (CAD); for further details see those entries. Even those methods do not really guarantee success. Formal methods succeed only if someone has a correct understanding of the problem to be solved, and successfully represents that understanding in a mathematical formulation. CAD allows a programmer to convert the mathematical formulation into usable software without error, because the human programmer provides only the intelligence required to decide how to proceed; the computer itself performs the actual steps of programming, and refuses to perform any step that does not conform to provably correct pre-established rules. Unfortunately, somebody has to create the CAD software in the first place, and, if the CAD application has a single bug in it, then every program it helps to create could be wrong.

Human beings make mistakes, and can not be prevented from doing so. Computers can not understand the requirements for software projects, and can not program themselves. Testing is not an adequate solution to the problem. Of course, testing is essential, but it is not sufficient. Testing can not possibly cover every case that may come up. All major software manufacturers have extensive testing facilities; the fact that bugs are still common is proof that testing does not solve the problem. Good programming practice can certainly drastically reduce the number of bugs in programs, but their elimination is not a practical possibility.

It is claimed that the term "bug" originated in the early days of computing, when engineers investigating a computer failure found an actual dead bug (in the sense of an insect) stuck somewhere in the works and causing the problem. This is not necessarily a true story, but it is widely believed and often repeated.

2: Remote surveillance devices in computing take two forms.

(1) Software illicitly installed on a computer that accesses stored data or records activities, transmitting records to the installer or keeping them for later retrieval. *Spyware* (q.v.) is an example of a software bug, but much more sophisticated forms exist. Good anti-virus software will find many instances of software bugs, and Adware removal tools find some others, but in critical situations where high-value data is at stake, there are no tools that may be completely relied upon. A trusted technician, fully familiar with exactly what software is supposed to be on a computer, and performing frequent regular checks, is a good but expensive precaution. The strict use of a *Firewall* (connection blocking software or hardware) also helps a lot, but nothing offers complete protection. It is quite possible (although certainly unlikely) that spying utilities could be built into the operating system or even the firewall software itself.

(2) Physical devices (the "bugs" of spy films) attached to the computer, which may transmit information by radio or by subverting existing channels (such as a network connection), or simply store information internally until the device can be retrieved by the planter. Kits are available for "sweeping" work areas and detecting any radio-frequency emissions, but they are of limited reliability, since bugs do not have to transmit all of the time, there are many frequencies that carry untraceable electrical noise, and bugs are not limited to radio transmissions. Again, frequent visits from a trusted technician who knows exactly what should be inside the computer will help,

but only partly. It is quite possible for hardware manufacturers to build perhaps disk drives or even CPUs with recording equipment embedded in their design. That is not to say that it does happen, but it is certainly possible.

Business value proposition

1: The term bug has several uses within the technology community, the most common being the bugs or errors in a program code. A famous example of bugs in the code is the "blue screen of death" that occurs when a personal computer crashes. Errors in software code are made by the programmers during the development process. Errors occur anywhere the code does not satisfy the specification or where the specification is in fact wrong and the code manifests the specification errors. Typically both cases are referred to as bugs. There are steps that can be taken to eradicate bugs and errors through the use of validation ("are we building the right product?") and verification ("are we building the product right?") techniques. Programmers sometimes have to examine thousands of lines of code to locate the error; however, software products known as debuggers are available to assist them in this task.

2: *Bugs* can also refer to surveillance devices. In situations where secrecy is required, countermeasures may need to be in place to overcome these issues. A bug may be of the cold-war variety with microphones in walls and telephones, but more likely in a modern context is invasive software known as *Spyware*. Key-stroke capturing software could be placed on a vulnerable computer, and the use of software to break into data repositories containing important intellectual property is always a threat. Careful counter-surveillance measures need to be put into place, covering both physical and software threats, and this is the responsibility of the company's chief security officer and the CIO.

Summary of positive issues

Development methodologies and techniques are available to minimize the occurrence of errors in a software system; *Formal methods* are a prominent path. The use of verification and validation techniques also assists in the minimization of errors in systems. Software developers have tools such as debuggers available to assist them in locating errors in their code.

Summary of potentially negative issues

Errors and bugs are present in all but the most carefully developed systems, and even careful development is no guarantee against them. The use of formal methods throughout development would theoretically lead to perfect bug-free software, but formal methods require exceptional levels of specialist training, and available tools provide incomplete coverage of the software development lifecycle. Even with the most perfectly designed software, simple errors in typing (either of the code itself, or in later data entry) may cause an error that lies dormant and undetected for years.

Reference

- S. Murrell and R. Plant (1977). "A survey of tools for validation and verification 1985–1995," *Decision Support Systems*, Volume 21, No. 4.

Associated terminology: Formal methods, Reliability, Programming language.

Bus

Foundation Concept: CPU.

Definition: A common connector used to transfer data, commands, or signals between connected devices. A common pathway.

Overview

In computing, the word *Bus* has the same derivation as the behemoth of public transportation: *omnibus*, meaning "*for all*"

(Latin, dative plural of *omnis*). A bus is a single elongated pathway shared by a number of subsystems, and used by all for communications. The alternative design is to provide every subsystem with a direct connection to every other subsystem; this could produce a faster overall system, since different pairs of subsystems could communicate at the same time, but would be prohibitively expensive and complex to implement. A bus replaces a multitude of direct connections with one shared medium. Busses do require special design considerations, to make sure that two pairs of subsystems do not attempt to communicate on the same bus at the same time, but this is only of concern to computer hardware designers; the question of whether or not to have a bus is irrevocably pre-decided for everyone else.

Busses occur at three distinct levels in a computer system. Inside the CPU, busses are used to connect the individual parts (arithmetic units, sequencers, memory controllers, etc.) to make the whole CPU work as effectively as possible. The design of the bus inside a CPU is invisible to users of the computer, and it can be effectively judged only by measuring the actual delivered performance of the CPU in controlled tests or *Benchmarks* (q.v.)

The CPU and other major components are connected by another level of busses on the *Motherboard*. Again, there is very little choice available, motherboard designers provide the best bus they can, and it can not be changed. If a user takes a strong dislike to the currently favored bus design (currently PCI), they will have a hard job finding any alternatives. The favored bus design does change as technology improves. The original IBM PCs used a 16-bit bus called ISA (Industry Standard Architecture), which was improved by EISA (Extended ISA), and the short-lived Micro-Channel Architecture. VESA (Video Electronics Standards Association) was a popular standard for a few years, supporting high-performance video cards, but now PCI (Peripheral Component Interconnect) is the almost universal standard for PCs. IDE, EIDE, and ATA (Advanced Technology Attachment) are the commonly used busses for connecting disk drives to the motherboard.

Off the motherboard, and usually outside the computer, there is a third level of bus use. Instead of having a socket for every possible device that might ever be connected to a computer, have one socket that connects to a bus, and many different devices may in turn be connected to that bus. USB (Universal Serial Bus) is the best-known current bus system, and, although many devices may be connected to a single USB bus, it is so popular that computers usually have two or four USB sockets. SCSI (Small Computer System Interface) is another common standard, now waning in popularity, for high-speed external devices. See *Port* for more information on USB and SCSI.

Minicomputer and mainframe manufacturers are usually less concerned with compatibility with third-party equipment, so larger computers tend to have proprietary internal bus systems (which are often called *Backplanes*), such as *Unibus* and *Q-bus*, but there are also independent standards such as the IEEE-1014 *VMEbus*. Mainframes often use proprietary busses for peripheral interconnection, such as IBM's FICON (*Fiber connectivity*) and ESCON (Enterprise System Connection), which can provide communication at 100 Mbits per second over distances exceeding ten miles, and Digital's DSSI. There are also independent standards such as ANSI's FDDI (*Fiber Distributed Data Interface*).

Business continuity service provider

Foundation concepts: Information lifecycle management.

Definition: Business continuity service providers are organizations that perform technology-related disaster planning and recovery services.

Overview

Owing to the mission-critical nature of technology within organizations, it is imperative for them to undertake disaster recovery assessments and planning. *Business continuity planning* (BCP) commences with a risk assessment of the current implementation and covers all aspects of the IT organization, assessing potential threats to the integrity of the organization and its systems. Risks include power failures, security breaches, failure of data backup systems, hardware failure, environmental threats (e.g., hurricanes), and infrastructure threats. The second stage of BCP involves managing the risk and developing responses to the threats through such means as backup power systems, UPSs, firewalls, improved physical and software security measures, redundant hardware, duplicate facilities (e.g., in environments not threatened by extreme weather), and stronger infrastructures. The use of stress analysis upon systems is a useful mechanism employed to show the potential for weakness in the systems.

The assessment of risk levels and the planning processes surrounding this assessment may be attuned to the business risk assessment. Should the threat be deemed high, an assessment of critical systems and emergency-level responses may be made for all technologies, people, and processes critical to the functioning of the business.

Business value proposition

The use of BCP allows businesses to consider a variety of risks and threats that could impact their business. The business continuity service providers (BCSPs) are third-party specialists who develop contingency plans for businesses. As service providers they act independently to assess the risks associated with a wide variety of potential threats and can work with entities to provide documentation, training, resources, and processes to ensure continuity of service.

Summary of positive issues

BCSPs provide independent assessments of risks and threats to an organization. They are capable of presenting current best-in-class process and technical solutions to corporations.

Summary of potentially negative issues

The development of BCP requires frequent updating as threats and risks change. The processes, people, knowledge, key documents, technology, infrastructure, customers, vendors, and other "critical resources" associated with corporations continuously change and this also forces BCP on a continuous basis.

Reference

- H. Wang (2004). "Contingency planning: emergency preparedness for terrorist attacks," *IEEE Aerospace and Electronic Systems Magazine*, Volume 19, Issue 3.

Associated terminology: Hosting, Backup, Information Technology Infrastructure Library.

Business intelligence (BI)

Foundation concept: ERP.
Definition: The use of analytic software applications to monitor business processes.

Overview

Business intelligence or *Business analytics* software helps corporations to perform analysis of their business processes. Business intelligence systems operate upon data that has been extracted, transformed, and loaded into specialist databases from online operational databases such as those found in ERP systems. These databases are typically referred to as *Data warehouses* and,

depending upon the architecture of the business intelligence system, the data warehouse may be a single *Enterprise data warehouse* that contains within it specialist *Data marts*, e.g., a finance data mart, a customer relationship management (CRM) data mart and a human capital management (HCM) data mart, or alternatively individual specialist data marts may have been created from, and exist outside of, the main enterprise data warehouse.

Business intelligence systems create an environment in which a process or set of processes may be carefully monitored. Data is drawn from the operational databases as required (e.g., every day or every hour) and placed into the data marts. The software then uses the data from one or more data marts, depending upon the architectural design, to generate reports. For example, a business intelligence system may be used to monitor a call center, providing the management with a visual "dashboard" through which to monitor such parameters as call volume, response time, length of call, and average number of calls per hour. Business intelligence systems simplify the production of reports.

Business value proposition

Business intelligence systems provide managers with the ability to monitor processes in a way that traditional transaction systems and reports can not. Business intelligence software is developed by vendors who can provide industry-specific solutions and tools that offer managers insights into their core business processes. The systems are configured to match the processes of the organization deploying them and are typically highly graphical in nature, depicting process metrics through a variety of mechanisms such as graphs, bar charts, and traffic lights. Depending on the design of the system, the software may provide *drill-down* capabilities for the user. For example, a system monitoring calls to a call center may display a map of the whole country, showing call volumes from each state. The user could then drill down to see the volumes from the counties within a state, then the cities, the zip codes and so on. At each level data relating to process metrics will be available to the user. Business intelligence systems also allow score cards to be created for processes and sets of processes based upon performance targets and the actual results as they relate to those metrics. The aim of business intelligence software is to help managers to make faster, better decisions based upon accurate, easy-to-interpret, and timely data.

Summary of positive issues

Business intelligence has emerged as a mature branch of the software industry and many vendors offer industry-specific solutions for a variety of processes within an industry or function. Business intelligence systems can usually be built onto an existing data mart or warehouse. The systems are intended to be highly graphical in nature and possess powerful user interface capabilities.

Summary of potentially negative issues

The deployment and use of a business intelligence system relies upon the availability of clean, well-structured data. Failure to create a well-formed data warehouse or data mart will invalidate any results presented by the system. Selection of the correct analytic tools and metrics is vital, and careful planning needs to be undertaken prior to the creation of the data warehouse and the business intelligence solution.

Reference

- E. Vitt, M. Luckevich, and S. Misner (2002). *Business Intelligence* (Redmond, WA, Microsoft Press).

Associated terminology: Database, Data warehouse.

Business process re-engineering

Foundation concept: Management information systems.

Definition: Business process re-engineering refers to the modification of one or more business processes in order to achieve a new process goal.

Overview

Business process re-engineering (BPR) is the task of examining a business process and redefining it to be more efficient, to meet changing business requirements, or to be more cost-effective. The re-engineering of a process involves many components, including the human capital, the physical items (if any) involved in a task, the information flow, and the resource requirements. Each of these needs to be modeled and the consequences of the proposed changes understood in order for a process to be modified successfully.

Processes have been a part of every business for centuries, but the advent of commercial computing in the 1960s enabled a wide range of business processes to be automated. *Data processing*, as it was then termed, led to the direct automation of existing business processes almost exactly as they were performed by humans. The systems developed were typically performed for a single process (e.g., payroll), and were termed *stand-alone systems*. Since they were not designed to communicate directly with other systems, they were also frequently called *islands of automation*.

The computerization of business processes helped to make industry aware that quantifiable, specifiable, and analyzable processes really were in place. During the 1970s systems managers worked to integrate (re-engineer) stand-alone systems, in essence creating bridges linking the islands. During this period systems analysts and programmers continued to attempt to make modifications to existing systems in order to accommodate changes in process requirements. This maintenance was frequently very resource-intensive and difficult.

In the 1980s, as the value of using information systems as a strategic tool became more apparent, it became increasingly important to fully re-engineer processes, linking together more systems, ranging from warehouse management to human resource management. However, the challenge involved in doing this with old legacy systems was becoming more difficult, as maintaining them from a technological point of view and to meet business process needs continued to drain corporate resources.

An answer to the problem of integrating legacy systems began to emerge through a new type of computing environment, known as *Enterprise resource planning* (ERP). These systems integrated many core business processes, and had only one database rather than many separate ones as was common with old Cobol-based legacy systems. ERP system vendors implemented best-of-class business processes within their software, typically specific to an industry vertical such as healthcare, retail, or manufacturing, and released the organization from the need to identify, design, and implement these processes. The advent of ERP systems with many configurable process options built into them allowed businesses to re-engineer their processes in significant ways. The best practices developed by the vendors allowed businesses to focus upon their core process competencies rather than worrying about upgrading and maintaining non-core processes.

Business value proposition

The philosophy underlying BPR is that businesses need to reconsider their processes from the physical and information-flow perspectives and re-engineer those processes that do not add value. Michael Hammer in his landmark 1990 *Harvard Business Review* paper "Re-engineering work: don't automate, obliterate" stressed the

importance of radical redesign rather than continuing to incrementally change processes that had originally been created decades ago for a different technological world. Re-evaluating and re-engineering processes can, if performed correctly, create smoother, faster, more flexible, lower-cost end-to-end business processes such as order-to-cash, procure-to-pay, and plan-to-report procedures. These end-to-end process cycles are typically supported by ERP systems and many large re-engineering efforts are thus centered on implementing an ERP and the best-in-class process model that it brings with it.

Summary of positive issues

BPR may be undertaken in conjunction with a systems re-engineering project, usually in the form of an ERP. ERP systems are mature and supported by consultants and vendors who offer solutions for companies of all sizes. BPR can enable organizations to determine what is core and what is non-core to their business, and transition to vendor-supported systems that manage their non-core processes. This allows an organization to focus upon its core activities.

Summary of potentially negative issues

BPR requires that organizations understand the implications of change. There is a potential for serious negative consequences if the BPR effort is performed badly. Unsuccessful ERP implementations that have been central to BPR efforts have in the past caused organizations to go out of business (FoxMeyer, a drug distribution company, suffered such a fate). BPR efforts require careful implementation because the workforce may be hostile to process change and work against the implementation of new systems and processes. Careful expectation management is required.

Reference

M. Hammer (1990). "Re-engineering work: don't automate, obliterate," *Harvard Business Review*, July–August.

Associated terminology: UML, ERP, Data-flow diagram.

C, C++, C#

Foundation concept: Programming language.
Definition: Standard, general-purpose programming languages.

Overview

The evolution of programming languages is a short but very complex story. By the mid 1970s the design of mainstream programming languages had branched in two directions: the general-purpose programming languages and the systems programming languages. General-purpose languages were strongly oriented towards supporting reliable software development; the structure of the languages corresponded closely to the structure of a good top-down design, providing programmers with a built-in framework for safe and verified construction; they were able to provide useful warnings about some of the most common things that could go wrong in a program's design before it is ever run, and they made many of the most troublesome of programmer errors impossible.

Unfortunately, what was good for general software development was not always good for low-level, or systems, programming. Implementors of operating systems, compilers, and hardware-monitoring software sometimes have to perform exactly the kind of low-level machine-oriented operations that are so strongly discouraged by the standards of good software development, and programming languages that enforce those standards are made almost unusable for such purposes.

As a result, the systems programming language branch of parallel evolution was also constructed. The essential feature of a systems programming language is that a programmer should be able to give any commands that the computer is capable of obeying, and not have to fight against built-in safety features if those commands do not satisfy some safety standard. The programmer takes full responsibility for ensuring the correctness of a program.

C was designed between 1969 and 1973, primarily by Dennis Ritchie of Bell Labs. It draws heavily from the design of an earlier language called B, which itself draws heavily from an earlier language called BCPL, which itself draws heavily from an even earlier language called CPL. When viewed in the context of systems programming languages, C was clearly a major positive evolutionary step. It enabled programmers to benefit from many of the lessons learned in the development of general-purpose programming languages without strictly enforcing any rules and without restricting what could be done. C became a convenient and much safer alternative to assembly language programming. Early on, C was adopted as the official language of Unix systems: all programming for Unix was in C, and every Unix system came with a standard C compiler at no extra charge. This undoubtedly fueled its increasing popularity.

No matter how good a tool is, it becomes a liability when used for the wrong purposes. The popularity of C for systems programming soon leaked into other areas, and it soon became arguably the most popular language for all programming tasks. The inability of C compilers to check for the most basic of errors in usage results in incredible numbers of programmer errors that simply could not be made in other languages. Inexperienced programmers exploit the flexibility of the rules, producing incomprehensible and unstructured programs, and delight in the clever but dangerous tricks that are possible. The *buffer-over-run problem*, which is still one of the most used vectors for virus delivery and breaking in to supposedly secure systems, may be directly traced to the design of C.

In the 1980s, some improvements and additions were made to the design of C, which resulted in there being two versions of the language. The original is known as *K&R C*, in honor of the authors (Kernighan and Ritchie) of the standard reference. K&R C is now obsolete. The new version was eventually adopted as an international standard (ISO/IEC 9899), and is now generally known as *ANSI C*.

Once C had been adopted as a general-purpose programming language, it made sense to put back into the language some of the features that had been deliberately left out when it was designed as a systems programming language, and to include new features that were not then part of the established paradigm for programming. C++ (pronounced "See Plus Plus") is a major extension of C, which was designed by Bjarne Stroustrup starting as early as 1979. It too has been adopted as an international standard (ISO/IEC 14882). C++ adds object-oriented programming and a vast library of predefined algorithms (known as the STL, or Standard Template Library) as well as a number of less major features. It also reasserts many of the safety rules that are inherent in general-purpose languages, but tend to be left out of systems languages. However, adherence to the rules remains optional.

C++ is an exceptionally large and complex language with many arcane rules. Even the most experienced of programmers are frequently surprised by the details. There was some negative reaction, founded on the reasonable notion that, if the language is so complex that it even has surprises for experienced experts, what chance have normal programmers of getting things right? But C++ continued undaunted, and has easily overtaken C in popularity.

Following on the popularity of Java, which provides the full power of object-oriented programming without all of the complexity of C++, and has proved to be another exceptionally popular programming language, today rivaling C++, the Microsoft corporation introduced C# (pronounced "See Sharp," like the musical note). It is essentially a fusion of C++ and Java. Like Java, it provides better safety both for programmers and for users than does C++, automates many of the difficult programming tasks (such as memory management), and provides an extensive library of predefined operations useful for general and network programming. C# makes many of the minor design decisions differently from Java, but is seen by programmers as being a very similar tool to work with.

Business value proposition

The major benefit of C is that it is a powerful language that can be used at a variety of levels, from systems programming to application development. For many major systems developments C has become the de facto standard programming language. For example, several ERP-systems developers have re-written their code into C, allowing their systems to be more easily modified than would have been possible when the systems were written in a proprietary or older procedural programming language. Use of the C language in these large systems developments has also encouraged application developers to create complementary applications, since the programmers do not have to learn a new programming language, and the use of a common language between applications reduces integration issues and problems. The use of C in conjunction with Unix also provides positive benefits due to their compatible design features.

The popularity of C has led to a large programmer base from which organizations can draw. Numerous compilers and tools have been created to support programmers, along with an enormous amount of

industry knowledge on the programming language and the techniques required to solve problems.

A similar situation has occurred with C++ and to a certain extent C#, both of which are used by programmers to write object-oriented code. These two languages are seen as a natural expansion of a programmer's skill set and thus programmers tend to be multi-lingual in C and its variants. C++ is also popular with developers because there is a large library of functions that save the developers from having to "reinvent the wheel," creating code that thousands of programmers may have created many times before.

Summary of positive issues

C, C++, and C# are languages with a large developer base. Developers using these languages are supported by a wide array of tools, compilers, and industry knowledge. C is designed to work in close unity with the Unix operating system. C++ has an extensive library of functionality from which a programmer can draw.

Summary of potentially negative issues

The inappropriate use of C can lead to poorly designed programs that may be very difficult to validate. C++ is a complex language that requires considerable effort to learn and completely understand. C# and C++ are not identical and the differences need to be clearly understood.

References

- B. Kernighan and D. Ritchie (1988). *The C Programming Language*, 2nd edn. (Englewood Cliffs, NJ, Prentice-Hall).
- B. Stroustrup (1997). *The C++ Programming Language* (New York, Addison-Wesley).
- http://msdn.microsoft.com/vcsharp/programming/language/.

Associated terminology: Java, Unix.

Cable Communications Policy Act of

Foundation concept: Data.

Definition: The US Cable Communications Policy Act of 1984 is intended to protect personal information related to cable service subscribers that is collected by cable service providers.

Overview

The US Cable Communication Policy Act of 1984 (CCPA) (Title 47, Chapter 5, Subchapter V-A, Part IV, Section 551, 1984) aims to protect the privacy of cable service subscribers. The code covers the obligations of the service provider; what data can be collected; what data can be disclosed; the rights of subscribers to access their own data; the cable providers' obligations pertaining to the destruction of data; guidance on civil action damages, costs, and fees; regulations pertaining to actions by states or franchising authorities; and disclosure issues pertaining to governmental entities.

Business value proposition

The act aims to limit the unauthorized collection of personal data that could be connected to an individual. Although it was originally intended to cover the collection of data from television cable subscribers, the act also relates to "other services provided by the cable operator" and explicitly identifies categories such as "any wire or radio communication service" and categories such as broadband internet service provision could be perceived as falling within the provision of this act.

Summary of positive issues

The right of an individual to data privacy is protected. Providers can apply encryption technologies to any data stored pertaining to an individual to prevent an unauthorized security violation.

Summary of potentially negative issues

It may be difficult for an individual to access or identify a carrier that offers

acceptable terms of data protection in the subscriber contract since all providers may require detailed personal information from those wishing to subscribe to their service, e.g., the individual's social-security number, telephone number, bank-account details, and driver's license details.

Reference

- Cable Communications Policy Act 47 USC §551.

Associated terminology: Security, Encryption.

Cables and connectors

Definition: A flexible physical connection that acts as a conduit for power, control signals, or data.

Overview

In the early days of computing and electronic communications, data processing and transmission speeds were so slow that the components of a computer could be connected together with simple insulated copper wires without any loss of performance. In modern computing, the design of cables adequate for high-speed data transfer has become a science in itself.

When a rapidly changing electrical signal flows along a conductor, such as a wire, some of its energy is actually transmitted into the environment as electromagnetic radiation (that is exactly how radio transmitters work). The more rapid the signal, the more of it gets transmitted. Conversely, when a conductor passes through an electromagnetic radiation field, part of that field is picked up, and converted to electric current in the conductor (that is how radio receivers work). Modern office equipment, especially computers, emits electromagnetic radiation on a wide spectrum of frequencies, and those very same emissions are picked up by that same equipment as interference that can be strong enough to mask the true operating signals.

Inside a desktop computer, the shielding provided by the metal casing is usually enough to protect the internal wiring, so long as that wiring is not unduly long. For example, it is not unusual for technically minded computer owners to buy extra-length IDE (motherboard-to-disk) cables so that their disk drives may be placed more conveniently. The extra length picks up extra interference, and can have a serious effect on system reliability. Mainframe computers with larger enclosures usually have very carefully designed internal shielding, which should not be interfered with.

Outside of the computer, cables for high-speed communications must be properly shielded; that is (part of the reason) why SCSI and USB-2 connectors seem to be so unreasonably expensive. Shielding may take the form of a grounded metal sheath completely surrounding the conductors (as in coaxial cable), or may be provided by carefully pairing signal-carrying and grounded conductors (as in twisted pair), or some combination of the two. The connectors at the ends of these cables usually provide some kind of metal fixing mechanism; this is not just to hold the cables in place, but also to provide continuity in the grounding.

Non-electronic communications media do not suffer from electronic interference. The prime example is *Fiber optics*, in which the conducting wires are replaced by transparent filaments, and signals are sent along them as rapidly modulated flashes of light, somewhat in the style of Morse code flashed on an Aldis lamp. Complete immunity to electronic interference means that fiber optics may be used in environments where electrical connections will not work, but the equipment required to send and receive signals along fiber optics is much more expensive. Of course, light itself would be interference for a fiber-optic cable, but the cable is easily shielded by providing an opaque covering.

Breaks and kinks in high-speed cables are also much more trouble. At low speed, a break in a conductor simply results in no signal getting through, an unfortunate condition, but one that is easily detected. At high speeds, the effect of self-interference can allow some attenuated portion of the signal to cross the break, making detection and repair more difficult. A kink in a cable can cause a short-span change in impedance (the cable's resistance to the flow of electricity); at low speeds, this simply reduces the signal strength slightly; at high speeds, it can cause reflections and echoes of the signal that drastically increase the overall error rate, and reduce throughput to a fraction of its expected value. With coaxial and twisted-pair cables (as used in broadband network connections), and especially with fiber-optic cable, it is essential never to kink or spindle the cable or force it into sharp turns, and, unless communications speed is not important, cables that have been kinked but still seem to work should probably be replaced. Carelessness with cables can be an expensive mistake in the long term.

Business value proposition

A few extra dollars for a well-designed, carefully manufactured cable or connector can easily save thousands of dollars in "troubleshooting" and repair costs, and incalculable amounts in saved down-time. Flimsy-looking metal covers should never be removed from computer parts; they may be too flimsy to provide any protection from physical damage, and it may seem sensible to increase ventilation, but metal covers provide essential electrical shielding.

Summary of positive issues

A wide range of cables with a variety of bandwidths and associated connectors is available.

Summary of potentially negative issues

The most common type of cable covering is PVC casing and, should it catch fire, the fumes can be hazardous. Many fire codes require that a special form of cable known as "Plenum rated" cable be used, which is fire-resistant. This is especially important if the cable is to be strung in the overhead spaces and recesses above offices and not enclosed in special ducting. Plenum cable is more expensive than PVC-cased cable and budgets need to be adjusted to take account of this expense.

Associated terminology: Ethernet, Bandwidth.

Cache

Foundation concepts: Storage, Memory, Disk.
Definition: A small but frequently accessed part of a data set, kept conveniently at hand for rapid access.

Overview

With any kind of data retrieval system, there are likely to be some data items that can be identified as more likely to be needed than others. For example, a corporate web server may store thousands of pages, any of which could be accessed at any time, but the corporation's "home page" and its search page will probably be accessed far more frequently than any others. When retrieval of data requires some time or expenditure of computational effort, it makes sense to set aside these most frequently accessed items in some special location that allows faster access if such a place exists. The set-aside store of frequently accessed entries is called a *Cache*.

In a system where normal access is through a network connection, a cache may sensibly be kept on a local disk. For example, a web browser usually keeps a cache of recently visited pages, so that pressing the "back" button does not require

any network access. In a system where normal access is to data on a disk, a cache is likely to be kept in memory. For example, a file-processing system may keep copies of recently accessed files in memory, because it is a common pattern to keep working on the same file until some task has been completed.

Web search engines often keep a cache of the results of popular searches, keeping dynamic statistics so that changes in popularity (as with people frequently accessing web sites that have been in the news recently) can be taken into account. A database system will often keep a cache of recently accessed records, or important indexes.

Caches also play an important part in hardware design. A disk drive controller will usually contain a few megabytes of memory, which is used as a cache to store frequently and recently accessed data, and, because memory accesses are so much faster than disk accesses, the presence of such a cache speeds up many common tasks by significant factors. Computer CPUs and motherboards usually contain a cache made of exceptionally fast memory, so that a similar speed-up can be achieved for normal memory accesses.

Business value proposition

The use of caches is an important mechanism through which a computer's abilities can be extended. A web server that stores popular corporate web pages in a cache will be faster than one that has to read repeatedly from backing storage; similarly, a computer that stores a user's previous web page access in a cache will give faster access to it than will one that doesn't. The cache is built into the system's design and, for software systems, it is usually an easy matter to reconfigure the size. Hardware caches are not normally at all flexible. It is a mistake to compare cache sizes for one CPU or disk drive with another, and conclude that the larger one is superior. Only a complete measurement of the system's speed in use will be meaningful.

Summary of positive issues

Caches allow computers to act more quickly than would otherwise be possible.

Summary of potentially negative issues

The use of a cache adds slightly to a system's complexity, but it is a well-understood technology, and is unlikely to have any significant negative effects.

Reference

- J. Handy (1998). *Cache Memory Book*, 2nd edn. (San Diego, CA, Academic Press).

Associated terminology: CPU, Bus.

CAD/CAM (computer aided design/

Definition: Computer aided design and computer aided manufacturing are two computer-based systems that provide a basis for product modeling and automated manufacturing.

Overview

Computer aided manufacturing (CAM) as we know it today evolved from early computers that used paper tape to issue instructions to numerically controlled devices (NCDs). An NCD is controlled by a machine-control unit (MCU), which is composed of two parts: a data-processing unit (DPU), which reads and processes the instructions in the control program, passing the instructions on to the other part, the control unit (CU), which converts those instructions into electro-mechanical control commands for the machine itself. In the early days this required special languages for each device because each manufacturer used a different instruction set. This problem was overcome when, in 1958, a language was

created by researchers at MIT to tackle it. The language, known as APT, allowed programmers to write in a standard code and then, through NCD-specific *Compilers*, transform the code into instructions that the NCD machine would understand.

The development of *Computer aided design* (CAD) is generally acknowledged to have started when Ivan Sutherland created a system known as *Sketchpad* at MIT in 1963. Sketchpad was an interactive graphics application that utilized one of the first graphical user interfaces and a *Light pen* to enable programmers to input designs. During the next twenty years the CAD research and vendor community developed more sophisticated applications to enable designers or programmers to create wire-frame representations of objects, view models as solid objects, rotate and scale objects, view cross sections of objects, etc.

As the CAD systems evolved it was natural for the output from CAD systems to be funneled into existing CAM systems, allowing the production and design issues to be examined. CAD/CAM systems have been used by organizations to design and manufacture nearly every type of product, with automotive and aeronautic systems being amongst the early adopters and driving forces behind much of the technological progress.

Business value proposition

CAD/CAM has the major advantage of allowing designers and production engineers to examine their models in virtual space without the intense commitment of resources and time that traditional methods require. These technologies are also tied in with other types of systems such as virtual reality (VR) modeling and computer aided styling (CAS) to enable designers to extend the design processes, both by viewing their systems in more realistic ways and with industry design-specific software. For example, the use of CAS in the automobile industry focuses on helping designers with the styling processes rather than the engineering processes that would underlie a design. These would be considered in the CAD stage of the development which follows the CAS stage.

The CAD/CAM systems allow organizations to capture their organizational knowledge within the systems and reproduce aspects of a design with low cost in future designs, thus avoiding the syndrome of "reinventing the wheel."

Summary of positive issues

CAD/CAM provides a fast, cost-effective mechanism to develop designs. The results of the system are precise and repeatable. The systems allow 3D images to be created, scaled, manipulated, and viewed. The systems can be linked to VR, CAS, and other tools used in the concept-to-production lifecycle.

Summary of potentially negative issues

The cost of the software itself together with the required training can be considerable. The integration of the components, the CAD and the CAM, needs to be carefully considered to ensure that the systems can communicate with each other using compatible, neutral data file formats such as the ANSI Initial Graphics Exchange Specification (IGES) standard.

Reference

- D. Schodek, M. Bechthold, J. Griggs, K. Kao, and M. Steinberg (2004). *Digital Design and Manufacturing: CAD/CAM Applications in Architecture and Design* (New York, John Wiley and Sons).

Associated terminology: Virtual reality.

Capacity planning

Definition: Capacity planning is the process of determining current and future resource requirements and developing cost-effective solutions to meet those needs.

Overview

Capacity planning can be considered as the mechanism through which systems managers develop their systems to meet future demands. Capacity planning has been in existence since the advent of corporate computing, when data processing managers would work to understand the performance capabilities of the mainframe systems that they were considering buying or leasing, matching that capability with a forecast of the company's future processing requirements.

The basis of capacity planning is the balance between two factors: cost of service delivery and capacity. Clearly, with an unlimited budget the capacity to satisfy almost any service level could be reached. Unfortunately, apart from special projects at government funded research agencies, few organizations have access to these levels of resources, so the task of capacity planning becomes a trade-off between the quality of the service delivered and the cost of the resources available that provide that service.

To undertake the capacity planning task, planners have two main tools: metrics and statistics. Metrics are used to measure past performance, for example network traffic may be measured in terms of bandwidth utilization over the entire network, historically 24 hours a day for a year. This would enable the network manager and capacity planner to determine statistically the average usage, absolute maximum usage, average maximum and average minimum usages, and other metrics. These statistics would enable a utilization analysis to be undertaken and a determination of current capacity limits to be established.

Having established the potential peak loads that could be placed upon the existing system, the future workload for the network may be projected and the next upgrade point established. Of course, it would be rare for one metric alone to be used in capacity planning since the problems are usually multivariate in nature. Metrics including quality of service (QoS) and end-to-end response time may also be important for a network manager in determining future peak load requirements. The overriding concern of the capacity planner is to ensure that the network does not run out of capacity prior to the next upgrade point.

Capacity planning can be undertaken for any systems component (physical or human), e.g., backup data storage capacity, server performance, help-desk service levels, and local storage in clients. Recently there has been much interest in the use of capacity planning techniques in the area of the web and web services. While the application of techniques and methodologies to these areas is fundamentally the same as measuring CPU loads on mainframes, in reality the complex nature of the technology that delivers the web services and their interactions makes capacity planning in these areas much more demanding.

Business value proposition

The development of a capacity planning model for an IT-based organization is one of the most critical projects that a CIO can undertake. The CIO must build the model in order to be able to present accurate capital budget requests. The CIO also needs to be able to look forward and backward in time. Looking back in time requires that the organization has established and enacted data collection programs that provide accurate data for desired metrics. These form the basis of the projections that CIOs and their staff make to determine future system utilization rates and project upgrade points. However, these upgrade points will be based upon a set of corporate operating assumptions. For example, projected system loads may be based upon a workforce growth of 1% per year for the next three years and the assumption that each user will have the same

systems-workload demands. Such projections would make sense in the case of a stable established company's IT help desk, but, if the company has a wide range of systems with highly demanding uses and is experiencing hyper-growth in employee numbers (such as an internet start-up company in the late 1990s), these conservative projections will be wrong and potentially harmful to the company.

Thus, CIOs require access to strategic decision making at the executive level of the organization. They can contribute to the process as well as obtaining planning and forecast information. The result is that better technology planning decisions can be made.

Summary of positive issues

Capacity planning enables organizations to provide a stable basis for their IT organization and provide a platform on which corporate strategy may be executed effectively. Capacity planning is a well-understood academic discipline that has substantial literature for practitioners and academics. Performance monitoring tools are built into many systems and can provide data for capacity planning tools built by specialty vendors as well as by the major software vendors.

Summary of potentially negative issues

The general metrics available from systems might not always be the most appropriate, so special metrics packages may need to be created or purchased. Demand forecasting can be complex, especially when data is limited or in newly emergent areas. Translating the impact of strategic decisions onto systems and developing forecasts from them is a difficult process and requires strong statistical and technical skills on the part of the capacity planner.

References

- T. Browning (1995). *Capacity Planning for Computer Systems* (New York, Academic Press).
- D. Menasce and V. Almeida (2002). *Capacity Planning for Web Services: Metrics, Models and Methods* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: CIO.

Cell computing

Also known as: **Grid computing**.
Foundation concept: CPU.
Definition: The use of a very large number of very weak, low-power interconnected computers instead of one fast, high-power one.

Overview

The research and development effort expended on producing ever more powerful computers is staggering, and staggeringly successful. Roughly every five years, top-of-the-line computers are ten times faster than they were before. There is no end in sight for this rapid development, but, as computers get bigger and faster, their power requirements increase disproportionately, design considerations become much more complex, and the chip fabrication process becomes more delicate. Naturally, alternatives are sought.

The idea behind cell computing is very simple: a hundred small, cheap, slow computer processors working together could easily outperform one very expensive, very fast computer. In simple terms, a CPU that was state-of-the-art five years ago now only has one tenth of the power of the current models, and as a result may be purchased very cheaply. Buying a hundred of them and making them work together could provide ten times the computing power of the fastest models available today.

Of course, the reality is not quite as simple as that. If it takes a person 40 minutes to drive to work in the morning, that does not mean that four people could drive to work in 10 minutes. Some tasks in computing, just as in real life, have an essentially serial nature, and can not be sped up

by parallel processing. However, many commercially useful computational tasks do have a very parallel nature. For example, if 1 000 000 000 un-indexed documents stored on disks are to be searched for those that contain some key phrase, the task would take a very long time, at least a day, and up to a month, depending on the sizes of the documents. Two computers could perform the task in exactly half the time; ten could do it in exactly one tenth of the time.

Some tasks have a very serial nature, some have a very parallel nature, and most are somewhere between the two extremes. Gene Amdahl, in 1967, analyzed the situation and the result is now known as "Amdahl's Law"; it is still the most frequently cited reference for the speed-up of processing obtainable through the use of multiple processors.

Cell computing really comes into its own at a different scale, not using a few dozen obsolete computers picked up cheaply, but using many thousands, perhaps even millions, of purpose-built computers. The Pentium 4 (introduced in 2001) has nearly 45 000 000 transistors on a single chip; the Intel 8080 microprocessor (introduced in 1974) had only 4500 transistors, but was still computationally useful. It is already possible to construct a single chip that contains 10 000 individual computationally useful computers.

Software development for cell computing would require major retooling in the industry. The ability to design a program for thousands of slow processors is a quite different skill from those normally possessed by software engineers. Unusually, the area has been very well investigated; C. A. R. Hoare of Oxford University thoroughly developed the computational theory in a series of papers on communicating sequential processes (CSP), culminating in his influential 1985 book. The programming language Occam implements the essential parts of CSP and is available for most programming platforms.

In the 1980s, the semiconductor manufacturer Inmos, in conjunction with the designers of CSP and Occam, produced and marketed a self-contained single-chip computer, called the Transputer, which was designed to support cell computing systems. Although it was technically successful, the Transputer never approached the low prices that full-scale cell computing requires, and was not a commercial success.

Wolfram claims to have developed a whole new basis for science, built on the ideas of cell computing, in his self-published 2002 book; the book is to say the least controversial, but does add to the development of the subject. These ideas grew originally from a computer game/simulation known as Conway's Game of Life, which was invented by the mathematician J. H. Conway in 1970.

The practical implementation of cell computing would have immediate application to high-power graphics production (one computing cell responsible for every pixel of the screen), and a brief investigation of this possibility reveals the important fact that some programming problems actually become much easier if they are designed for large-scale cell computing instead of the traditional uniprocessor computers. It is also expected that cell computing would be very helpful in modeling physical phenomena, especially the atmospheric models that support weather forecasting. It is equally clear that the full benefits of cell-computing will not be fully known until cell-computing hardware is generally available.

It is known that the *Neuron*, the basic brain cell, has exceptionally simple functionality that is easy to duplicate with very inexpensive circuitry. A neuron is incapable of performing any calculation or harboring any thought alone; it is only through the vast number (over 100 000 000 000) of neurons in the brain, and their complex interconnectivity, that intelligent thought is

possible. The possibility of being able to fabricate a brain's worth of artificial neurons is still in the very distant future, but the undeniable success of real neurons fuels copious academic research toward what is perhaps the ultimate application of cell computing.

Recently there has been some revival of interest in cell computing in the mainstream of the computing industry, in the form of IBM's CELL project, or broadband processor architecture. This has been fueled by the desire for ever more realistic graphics in video game consoles, which are amongst the most computationally demanding computer applications in common use.

Business value proposition

The commercial potential for cell computing is undeniable, but the current reality of the technology is that it lags considerably behind theory. Industry has for the last few decades pursued the goal of putting more and more transistors onto a single chip rather than working on simpler chips that can communicate with each other in large numbers. As the limits of technology are reached, it will become more important for the extremely high density chips to communicate with each other and subdivide the tasks to which they are set in order to continue to make gains in performance. The move to cellular systems will also require a major change in the style of programming and the development of new software systems to support the technology and its environment.

Summary of positive issues

The theory of communicating sequential processing is well known and understood in the computer science research community. Cell computing has been developed in the past and applications based upon those technologies were developed. Amdahl's Law governs the processing performances obtainable through the use of multiple processors programmed in traditional ways. Several vendors are working on and deploying new forms of cell technology.

Summary of potentially negative issues

The use of large-scale communicating processes using a cell processing model is largely confined to the research laboratory and will require extensive research to bring the technologies into the commercial realm.

References

- G. Amdahl (1967). The validity of the single processor approach to achieving large-scale computing capabilities," in *Proceedings AFIPS National Computer Conference* (Anaheim, CA, AFIPS), pp. 483–485.
- C. A. R. Hoare (1985). *Communicating Sequential Processes* (Englewood Cliffs, NJ, Prentice-Hall).
- G. Jones (1987). *Programming in Occam* (Upper Saddle River, NJ, Prentice-Hall).
- S. Wolfram (2002). *A New Kind of Science* (Champaign, IL, Wolfram Media).

Associated terminology: Parallel processing.

Chaos theory

Definition: A chaotic system is one that may have broad areas of stable predictable behavior, but also has areas in which immeasurably small changes in the initial conditions produce significant changes in the results, thus making long-term predictions impossible.

Overview

Contrary to beliefs current in popular culture (*Jurassic Park* to be precise), chaos theory does not predict that complex systems will fail. It has absolutely nothing to say on the subject.

The truth about chaos theory is much more interesting and surprising. Since

Newton postulated the basic laws of physics in the mid seventeenth century, people have generally believed that the universe works like a big machine: everything that happens is the direct result of other things that happened before. If you know how the machine is initially set up, you can predict exactly what it will do in the future.

Weather forecasting is a good example. The underlying science of the weather is well known: how a small volume of air, land, or sea behaves when temperature, pressure, or other attributes vary, and how neighboring small volumes of air, land, or sea interact with each other are all understood. Until quite recently it was believed that if data could be gathered, accurately measuring the current state of the entire atmosphere, a sufficiently powerful computer could use the known equations of physics to determine how the system would evolve over time, and produce accurate long-range weather forecasts.

It was realized from the beginning that imperfect measurements would produce imperfect forecasts, but it was quite reasonably expected that good-enough measurements would produce good-enough forecasts, and gradual improvements in measurement technology would produce corresponding improvements in forecast accuracy.

It is now known that this is not the case. In many scenarios it does work out quite well: small errors in measurements produce only small errors in the forecast, allowing rain and drought to be predicted; these are known as *stable* conditions. Unfortunately, unstable conditions abound. Consider the formation of a hurricane: there are conditions under which a hurricane will form, and there are conditions under which it won't. Between those two sets of conditions, what happens? Initial conditions are infinitely variable, but a hurricane either forms or it doesn't. There is some point at which the tiniest immeasurably small change in conditions makes the difference. These *catastrophe points* always exist; no matter how accurately conditions are measured, there are always transitional points at which that accuracy is not enough to predict correctly a major yes-or-no event. If a forecast says "no hurricane" when there is a hurricane, then, of course, everything else is going to be completely wrong too.

It is generally accepted that other systems of a similar nature, namely interacting components with conditions that can not be perfectly controlled or measured, can be expected to exhibit chaotic behavior and resist long-term forecasts. The behavior of human or animal populations and stock-market prices are prime examples.

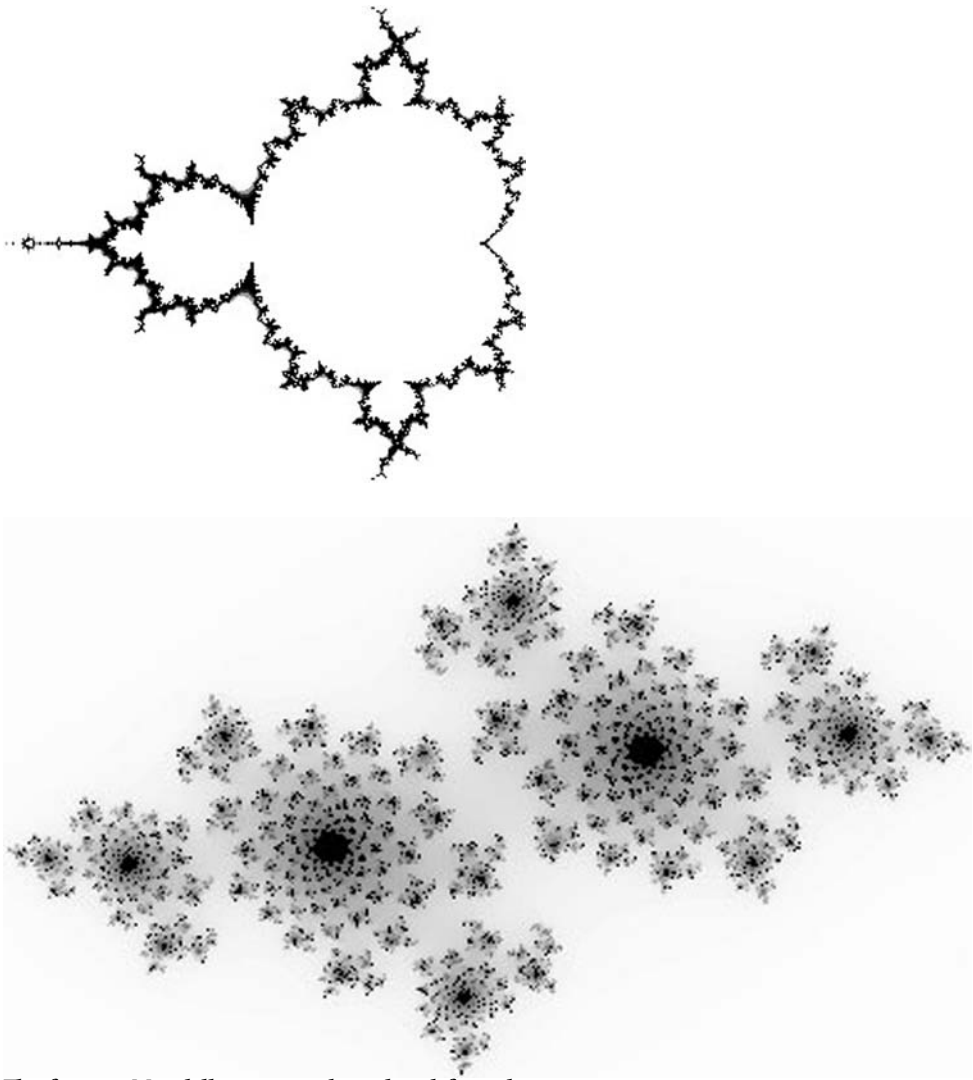
The fine boundary between sets of conditions with drastically different outcomes is known to science as the Julia set (after Gaston Julia, French mathematician, 1893–1978), and is often a fractal (grossly simplified, a fractal is a complex shape whose length can not be measured). The most famous of these is the Mandelbrot set, which is produced from a very simple mathematical formula, $x' = x^2 + c$, and demonstrates that even exceptionally simple, completely known mathematical equations can exhibit chaotic behavior.

Business value proposition

The study of chaos theory and fractals has led to the abandonment of projects now known to be infeasible, the ability to produce realistic graphical renderings of natural scenes from very little data, and the discovery of some exceptionally effective data compression methods.

Summary of positive issues

The study of chaos theory has revealed some fundamental truths of the universe and produced some useful techniques as a by-product, rather in the manner of NASA moon missions being credited with the



The famous Mandelbrot set and a related fractal.

development of new technologies that have found their ways into everyday life, except that there is a rather different level of expenditure involved.

Summary of potentially negative issues

Claims of the general applicability of chaos theory to problems such as modeling the stock market to predict future performance need to be considered carefully, since such models rarely produce results with high degrees of reliability. Research into the use of chaos theory and fractals is very demanding and theoretical, and is unlikely to be successfully carried out by anyone lacking a Ph.D. in mathematics and a great deal of time.

References

- B. Mandelbrot and R. Hudson (2004). *The Misbehavior of Markets: A Fractal View of Risk, Ruin and Reward* (Cambridge, MA, Perseus).

- M. Barnsley (2001). *Fractals Everywhere* (San Diego, CA, Elsevier).
- G. Williams (1997). *Chaos Theory Tamed* (Washington, DC, Joseph Henry Press).

Children's Online Privacy Protection

Foundation concept: Security.
Definition: The US Children's Online Privacy Protection Act of 1998 is intended to "prohibit unfair or deceptive acts or practices in the collection, use, or disclosure of personally identifiable information from and about children on the internet."

Overview

The US Children's Online Privacy Protection Act of 1998 (COPPA) (Title XIII, Section 1301, 1998) defines the privacy policies to which "web sites or online services" aimed at children under 13 must adhere. Key provisions of the act are the placement of prominent policy notices on web sites; the notification of parents that a site is going to collect personal information; the mechanisms for obtaining parental consent prior to a site obtaining or collecting personal data; to disallow collection of data as an incentive for further involvement in the site by the child; to allow parents to review information collected on their child and have that information deleted if they wish; to allow parents to prohibit an online service or web site from collecting any further data on their child; the role of schools in providing permission; the dissemination of data to third parties; and the procedures by which online services maintain and delete data.

Business value proposition

The act requires the operators of web sites aimed at children of 12 and under to operate within a defined framework.

Summary of positive issues

The act is comprehensive, defining the responsibilities of the service providers, the web-site operators, and the parents, and their interactions. The act aims not to be onerous and has special provision for cases of limited interaction (such as a site receiving an email from a child to which only a single response is needed). In such a case parental notice and consent are not required as long as all personal information obtained from the child is deleted upon execution of the email.

Summary of potentially negative issues

A problematic aspect of the act has been the mechanism through which "verifiable parental consent" is obtained. At the outset of the act the final rule devised a "sliding scale" that allowed the methods used by web sites to "be reasonably calculated in light of available technology, to ensure that the person providing consent is the child's parent." These methods include postal mail, "print-and-send" faxes, credit-card usage, toll-free verification, and digital signatures.

References

- The Federal Register, March 15, 2006, Part III, Federal Trade Commission, 16 CFR Part 312, Children's Online Privacy Protection Rule; Final Rule.
- 15 USC §§6501–6505 (2006).

Associated terminology: Law cross-reference.

CIO (chief information officer)

Foundation concept: MIS.
Definition: The chief information officer is a senior-level executive responsible for all corporate information systems.

Overview

The role of a *Chief Information Officer* (CIO) is to ensure that the systems and technology within a company are aligned to the company's strategy and that all systems are compliant to government regulations. The

CIO is a member of the executive layer who typically reports to the CEO and the board of directors or through the office of the CFO.

The CIO as head of the IT organization is an integral part of the corporate strategic planning group and helps formulate corporate strategy by bringing an understanding of the role of technology within the company's marketplace, the impact that new technologies will have upon the industry in which the company operates, and the technology-related regulations that impact the environment in which the company operates (e.g., HIPAA, Sarbanes–Oxley).

CIOs manage departments that encompass complex business processes and technologies that support business processes. Issues with which the CIO has to contend include business process re-engineering, systems developments, maintaining legacy systems, training, technology assessment, developing contingency plans, ensuring legal compliance, technology deployments (ERP systems, robotic systems, etc.), technology outsourcing, and technologies that support specific functional aspects of the business (e.g., customer relationship management systems).

Business value proposition

The role of the CIO has changed considerably over time as IT organizations within companies have increased in visibility and value. MIS departments were, up until the mid 1980s, frequently considered as data processing operations, run as cost centers within the organization by data processing managers. Subsequently, MIS departments became recognized as value centers providing "enabling" technologies to the core business processes and offering the potential for competitive advantage to be gained through their strategic deployment. During this period data processing managers became replaced by CIOs, who began to view themselves not only as organizational enablers but also as organizational transformers. As such, a visionary CIO who can also execute technology solutions has become a major corporate asset as both technology and corporate strategies continue to evolve rapidly.

Reference

- M. Earl, M. Feeney, and D. Feeney (1994). "Is your CIO adding value?," *Sloan Management Review*, Volume 35, Issue 3.

Associated terminology: Sarbanes–Oxley Act.

Click-stream tracking

Foundation concept: e-Commerce.
Definition: Click-stream tracking systems are applications that monitor web site accesses and compile data based upon visitor activity.

Overview

A click stream defines a sequential list ("stream") of activities (also known as "clicks" or "hits") that an individual visitor performs on a web site. Click-stream tracking systems monitor the click streams, consolidating the data and generating reports.

The applications that monitor click streams range from those that come as part of an internet server package, to custom solutions from vendors who specialize in tracking and monitoring internet and network traffic. Systems typically offer a variety of data analysis tools to monitor a wide variety of parameters, including metrics such as the total number of visits to the site, total number of pages viewed, total number of hits to the site, total number of unique visitors, total number of new visitors, total number of repeat visitors, total usage number, average number of visits per day (week, month, etc.), time period of highest activity levels (hour, day, month, etc.), average length of visits, average number of times a site is visited by an individual (assuming that the IP address is known), most frequent visitor, domain, host, or user,

most popular content accessed, most popular paths, most frequent entry and exit points, most frequent keyword search, most popular links, most popular browser used to access the site, most popular platform, most popular screen resolution, error analysis, and server response analysis data.

Raw data on page-visit sequences is typically accumulated in session *cookies* (q.v.), which are updated and exchanged between browser and server on each access, and would not contain any personal data. Occasionally, hidden input elements and javascript methods may be used as an alternative implementation method.

Business value proposition

The click-stream analysis packages allow corporations to monitor web traffic profiles and reposition their online strategy accordingly. They can, for example, use applications that perform marketing analysis of their online channel. This can be performed by categorizing specific aspects of the web site as belonging to a particular product segment, e.g., toys, books, and videos. These can then be monitored and analysis performed to assess the impact of marketing campaigns, for example the impact of an advertisement in a particular search engine. Typical channel-based metrics would include keyword searching, hyperlink referrals (where the customer came from prior to entering the site), robot or spider activity, and customer technology profiling (e.g., their browser, platform, screen resolution, and ISP).

The more sophisticated analysis suites can monitor browsing patterns, keeping data on the sequence of pages viewed by site visitors. This information enables site designers to check the quality of their layouts and ensure that visitors are able to find what they are looking for without getting too annoyed. It can also help informational sites to identify bottlenecks, and group frequently co-visited pages more efficiently.

Summary of positive issues

Click-stream analysis is supported by a variety of application vendors and consultants. The data collected can be customized to a particular organization and its requirements, which can be a technical analysis of their site (e.g., performance-related) or marketing-related (e.g., which aspect of the site is popular and which is not).

Summary of potentially negative issues

Click-stream analysis is often perceived as related to spyware and other invasive software. However, when it is used in isolation this is not the case, since click-stream analysis is passive data collection and all activity occurs on the web server. Click-stream analysis is, however, frequently combined with data collected at the client or customer site through the use of less legitimate software.

Reference

- M. Sweiger, M. Madsen, J. Langston, and H. Lombard (2002). *Clickstream Data Warehousing* (New York, John Wiley and Sons).

Associated terminology: Cookies, Spyware.

Client

Foundation concepts: Network, Software.
Definition: Clients are computing devices through which end users interface with a network-accessible system. They do not in themselves provide network resources.

Overview

A client can be any device through which an end user interacts with a networked computing environment. An example of this type of device is a personal computer with a web browser that acts as a *front end* to the resources of a network. Through the client, the user interfaces with the network to download files, browse resources, or use

an application such as *instant messenger* to communicate with another end user. Clients request services from other devices on the network; those devices which deliver them are commonly called *Servers* (file servers, web servers, application servers, etc.).

Protocols, or intermediate languages of communication, are designed to enable clients to interface with servers regardless of any operating system or software incompatibilities between the two: a Windows client can successfully interact with a Unix or Macintosh server because all communications are through a system-independent protocol. Of course, it is still necessary to have a client that can handle the appropriate protocols; a web-browsing client is unlikely to be able to communicate with a database server, but the client/server division prevents a lot of potential problems. See *Client–server* for further details.

Business value proposition

The client concept allows corporate networks to interact with many different types of device, from a large powerful workstation on which business intelligence software resides, and to which data is regularly downloaded from the network for local analysis, to a personal digital-assistant-type device through which employees can gain access to email on a continuous basis. Clients allow greater flexibility of connectivity to corporate computing resources than does the traditional *Dumb terminal* configuration that previous generations of systems deployed. Clients allow individual end-user cost-of-ownership decisions to be made and increase the potential for a greater return on investment in system architectures.

Summary of positive issues

Clients facilitate flexibility of network architectures at the end-user level.

Summary of potentially negative issues

Failure to understand the total cost of ownership at the end-user level can lead to excessive recurring systems expenses and a great waste of resources.

References

- W. Stevens (1994). *TCP/IP Illustrated*, Volume 1 (New York, Addison-Wesley).
- R. Orfali, D. Harkey, and J. Edwards (1999). *Client/Server Survival Guide*, 3rd edn. (New York, John Wiley and Sons).
- J. Chellis, C. Perkins, and M. Strebe (2000). *MCSE: Networking Essentials, Study Guide*, 3rd edn. (Alameda, CA, Sybex Press).

Associated terminology: Server, Client–server.

Client–server

Foundation concepts: Client, Networks, Server.

Overview

When computers are connected together on a network, it is common practice to localize certain functions. Prior to the advent of local area networks and wide area networks, a typical business office environment would have encompassed a number of general purpose computers, or workstations, upon which would have been performed self-contained tasks (word processing, spread-sheet processing, etc.); should there have been a need to share files, they would have been transported via floppy disks. It soon became apparent that this was a limitation on workflow and that the ability to share files was central to organizational efficiency. This prompted the development of networking. The first evolution was in terms of a peer-to-peer configuration whereby everyone in an office would be connected to everyone else. This resulted in connectivity problems due to the complexity of the networks required and

the security relations between computers within that network. Thus it became clear that the computers within an organization needed to be connected through a centralized computer, the server, or a group of servers acting in a coordinated manner yet controlled by a central primary computer. Then, when co-workers need to share a file, the file in question is copied from the file server, over the network, onto one of the client workstations where it is to be used. If any modifications are made, the changes are sent back to the file server so that all other workers will be able to access the new version.

Similarly, when email arrives from outside the organization, there needs to be a designated computer that is always online and ready to receive it. When an associate wishes to read their email, their client workstation makes contact with the email server, and retrieves all relevant unread messages.

Having one computer (or more) specially designated as "the server" means that important files can be kept safe in one central location that is carefully protected, and there is always an online presence to receive email and handle web services, regardless of what the various workstations may be doing. Although the server is probably the most important of the computers, its role is mostly passive. Client–server architectures are also known as three-tier architectures from the fact that the clients (tier one) request information from the primary server (tier two) and they in turn request the service from the specialized servers (tier three), which provide a service such as keeping files, emails, or web pages; they respond only to requests from the primary server.

This is by far the most common organization for network services. When communications are made, it is because one computer or system was passively waiting for requests, and another actively made such a request. This is in direct contrast to the *peer-to-peer* organization, in which the two communicating systems have the same status. With a client–server system, it is important that the server(s) have known, relatively static locations (usually IP addresses), but clients can be very mobile.

The client–server relationship does not apply only to whole computers. Any given application that has network access may act as a server or as a client, or even both. On any computer at any time, there could be a number of applications acting as servers, passively waiting for requests, and a number of applications acting as clients, driven by the user actively making connections with servers. FTP is an example of an application that acts both as a server and as a client concurrently.

Business value proposition

The three-tier client–server architecture is by far the most common computing configuration found within organizations. It is predominantly centralized in nature, with flexibility of client location and access through the networking protocols utilized to connect the client to the server (TCP/IP). The network response can be improved on large-scale global client–server networks by defining groups of clients and having mirror-image or secondary domain controllers administer the users of those clients. The primary and secondary controllers cocoordinate in order to remain synchronized in terms of user profiles, security levels, file structures, etc. This ability to group users into domains and have localized control yet global access to resources has made this the standard choice for the majority of organizations because it simultaneously facilitates security and flexibility of the network.

Several variations on the nature of client–server architectures exist, two major categories of which are *Thin client* and *Fat client*. In thin-client architectures the client itself

is a computer with little or no local processing power, displaying only the images sent to it by the server and being capable of sending requests to the server input by the end user. Fat clients, on the other hand, are computing devices with processing power at the client side of the relationship. This may facilitate the client being able to process data rather than making the server process any and all data as would be the case in the thin-client architecture. Similarly, fat clients may also have applications residing upon them and this also allows local processing of data upon that application; thin clients would not have this facility. The value proposition for the business is that this architecture allows the organization to determine the total cost of ownership of their computing resources in a number of configurations and develop a best case for return on investment. Furthermore, the majority of large software systems such as ERP systems are now written for client–server architectures and vendors are withdrawing support for mainframe or stand-alone-based architectures.

Summary of positive issues

Client–server three-tier architectures facilitate flexibility of architecture design and map comfortably upon organizational structures. This type of architecture provides centralized control of the network, enhances its security, and is scalable. Software vendors use the client–server architecture as their standard model upon which they develop their software.

Summary of potentially negative issues

Client–server architecture requires extensive knowledge to manage effectively. Failure to understand the total cost of ownership at the end-user level can lead to very expensive recurring systems expense and a waste of resources.

References

- W. Stevens (1994). *TCP/IP Illustrated*, Volume 1 (New York, Addison-Wesley).
- R. Orfali, D. Harkey, and J. Edwards (1999). *Client/Server Survival Guide*, 3rd edn. (New York, John Wiley and Sons).
- J. Chellis, C. Perkins, and M. Strebe (2000). *MCSE: Networking Essentials, Study Guide*, 3rd edn. (Alameda, CA, Sybex Press).

Associated terminology: Network, Server, Client, TCP, Internet protocol.

Clone

Foundation concept: CPU.

Definition: A complete computer or just a CPU, built by an independent manufacturer, but conforming exactly to the specification of a "famous brand," usually the IBM PC or the Intel 80486, to allow direct substitution of the product.

Overview

The term "PC" was originally a completely generic term meaning "personal computer" without reference to any particular manufacturer. The term was in common use many years before IBM produced its first PC in 1981. However, IBM cleverly incorporated the letters PC into their product names ("IBM PC/AT," "IBM PC/XT," etc.), and since that moment the phrase PC has been taken almost universally to mean "personal computer designed by (or compatible with) IBM."

Once IBM's line of computers had achieved total market domination, it was inevitable that others would want a share of the vast profits available. At first, this resulted in many "third-party" software houses: companies that do not produce computers at all, only software or applications. Software written for one type of computer will generally not work at all on another type of computer, so software manufacturers concentrated their efforts on the most popular

platform: the IBM PC. IBM-PC-compatible software became as popular as IBM PCs.

This gave rise to the "*Clone*." A clone is a computer that is exactly compatible with an IBM PC, so completely compatible that it can run all software designed for an IBM PC without modification, and thus may be bought instead and substituted directly for the more well-known product.

The IBM-PC design is based on the Intel microprocessor architecture. Any CPU that behaves exactly as the corresponding Intel CPU would behave may be used in its place. Thus the second type of clone: a microprocessor CPU not made by Intel, but designed to be directly substitutable on PC motherboards. Many of the product lines that began as Intel clones have developed into totally independent designs that are no longer clones in the normal sense.

Business value proposition

The vast majority of computers today are clones, so much so that the term is losing its meaning. For a computer to be a clone is no longer a derogatory term. There are many variables to consider in selecting a CPU, and no single product or manufacturer can claim to be "the" best.

However, designing a whole computer is an expensive and complex undertaking. Poor design, especially of the motherboard, results in poor performance and unreliability. It is important to select a manufacturer that can put enough resources into proper development of their products. A good clue is to start reading the "user manual": if the text is of poor quality or incomprehensible this may be an indication of other quality issues associated with the computer itself, or its manufacturer.

Summary of positive issues

Competition provided by alternate manufacturers ensures healthy development budgets, continuing improvement of product lines, more options, and lower prices. For most desktop/workstation uses, the differences between manufacturers and CPU families are almost irrelevant, so, although the variety of choices may seem bewildering, it is not really a problem.

Summary of potentially negative issues

Many very cheaply made computers simply do not work properly.

Cluster

Foundation concepts: Server, Network, Disk.
Definitions:

1. *A group of computers configured to act as one.*
2. *A group of blocks on a disk.*
3. *A group of similar data items that may be related.*

Overviews

1. In situations where a single computer is powerful enough to support the needs of a moderate number of concurrent users, but not powerful enough to support the large number actually expected, it is common to use *clustering*. A *Cluster* is a group of computers, usually configured identically as servers. They will generally have the same software installed, and, either through disk-sharing or by providing each computer with its own copy, will have access to the same data.

The cluster appears to the outside world, under casual inspection, to be a single computer. A system of dynamic load balancing is used to redirect all user requests to the member of the cluster that is currently least occupied. Two users "logging in" at the same time may find themselves connected to different real computers, but still having access to all the same files. A cluster of servers is often called a *Server farm*.

With clusters, it is common to use a system of *Load balancing*. This is a software consideration that attempts to keep the processing load on each of the members of a cluster reasonably balanced.

Implementations range from the extremely simple method of assigning incoming simple requests to individual members in turn, in a "round-robin" fashion, so that they will be balanced on average, to having one system with special responsibility for monitoring the state of each of the others and actively redirecting requests to the least busy, or having each member provide continually updated assessments of its own load to a much simpler central redirection service. It is also possible under some circumstances to have computations that have already begun on one cluster member migrate to another when load conditions change, although this is a very complex procedure.

2. A *Block* is the smallest individually accessible unit of storage on a disk system; it is normally 512 bytes in size. Operating systems assign numbers to blocks to act as their addresses, and use the *Block number* as a way of finding data. Many older operating systems use only small-range numbers for identifying blocks (Windows 3.1 used numbers in the range 1–65 527), and this created an unacceptably low limit on the usable size of a disk (65 527 × 512 bytes is only 32 MB).

The solution was to consider groups of neighboring blocks to be one single *Superblock*, and to assign numbers to these superblocks rather than to individual real blocks. Super-blocks are properly called *Clusters*. If a disk drive were configured to have clusters of 64 blocks each, the maximum accessible size would be raised to 2 GB. The disadvantage is that one cluster becomes the smallest allocatable amount, and thus even a file containing only one byte of data would occupy 32 768 bytes of disk space.

3. In data analysis and *Data mining* (q.v.) a cluster is a group of data items that are more closely related to each other than can be expected from random chance, and are thus assumed to have some meaningful inter-relationship.

Business value proposition

Clustering can be a very cost-effective solution. Twenty normal computers often cost less than one super-computer of 20 times the power. More importantly, a cluster creates a very robust system: if one computer fails, the 19 others continue unaffected, and the system simply runs at 95% of its original power. Maintenance of both software and hardware may be performed on a rotating basis, one cluster member at a time, so there is never any need for downtime.

Summary of positive issues

The computers, networking, and other hardware used in clusters can be treated as "commercial off the shelf" (COTS), providing a low-cost architectural solution and an associated lower maintenance cost than a specialized hardware solution. Clusters can be created using proprietary operating systems as well as open-source systems. Clusters provide reliable and scalable solutions.

Summary of potentially negative issues

Clusters might not be suitable for all processing requirements, especially those requiring close connectivity or interactions amongst tasks. The bandwidth and response-time requirements of a high-volume transaction-processing system may be too great for the intra-cluster network, in which case a high-power mainframe computer may be necessary.

References

- K. Kopper (2005). *Linux Enterprise Cluster: Build a Highly Available Cluster with Commodity Hardware and Free Software* (San Francisco, CA, No Starch Press).
- T. Sterling, J. Salmon, D. Becker, and D. Savarese (1999). *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters* (Cambridge, MA, MIT Press).

Cobol

Foundation concepts: Programming language, Compiler.

Definition: The Common Business-Oriented Language (Cobol) was developed to facilitate the creation of business applications.

Overview

Cobol, a programming language proposed in 1952 by computer pioneer Admiral Grace Hopper, was formally specified in 1960 at the Conference on Data System Languages (CODASYL), and became an American National Standards Institute (ANSI) standard in 1968. The language was intended to provide an accessible mechanism through which business applications could be developed.

The language was designed to be easy for non-technical programmers to read and write, which was very much against the style of the day. Commands in Cobol appear to be plain English instructions, with a limited vocabulary and strangely stiff syntax, including sentences such as "SUBTRACT DISCOUNT FROM TOTAL," "SORT CUSTOMER-FILE ON ASCENDING KEY ACCT-NUMBER," and "PERFORM INPUT-PROCESS UNTIL FILE-COMPLETE."

Although the successful use of Cobol for programming does require substantial training, it did open up commercial software development to a new generation of programmers without scientific or engineering backgrounds. Cobol was very different from the primarily mathematically oriented languages that were available to developers in the 1970s such as Fortran, which were difficult to use for the development of business applications that involved a significant amount of file and record processing as opposed to computation. The emergence of Cobol by the 1970s and its rapid adoption by developers were also due in part to the emergent nature of more powerful online database technologies and file structures that Cobol could manipulate. From the 1970s until the late 1980s Cobol became the most widely used programming language for business applications and compilers were developed for a wide variety of platforms, ranging from large mainframes to personal computers. Cobol has continued to evolve and in 2002 both ISO and ANSI released new standards (ISO-1989).

Business value proposition

Cobol has been used for around 40 years in developing business applications, and there is a large body of knowledge and experience on the programming language, and many structured development methodologies have been created to support it.

Summary of positive issues

Cobol is a relatively easy language to learn and use, and in which to develop applications. Billions of lines of code have been written over the past 40 years and there is a deep pool of technical and human resources available to support applications and their development. Many CAD tools, structured design methods, test suites, and systems have been developed to support Cobol. Cobol is continuing to evolve and systems have been developed both to enable Cobol programs to be converted to other languages, such as Java, and to receive input from XML documents.

Summary of potentially negative issues

Cobol has been declining in use since well before the year 2000 when the "Y2K" problem was understood and technically solved. Y2K was the result of representing years in files by two digits (e.g., 1966 as 66), and consequently programs in Cobol (and other languages) reading and manipulating two-digit date fields. Owing to the length of time over which Cobol has been used, some programs may have become unstructured in nature over their lifespan as

they were amended by many generations of programmers. This leads to difficulties in understanding, modifying, and testing aged Cobol systems.

References

- D. McCracken (1970). *Guide to Cobol Programming* (New York, John Wiley and Sons).
- http://www.cobolstandard.info/wg4/standard.html.
- H. Hinman (2004). *Microsoft.NET for COBOL Programmers*, 2nd edn. (Sunnyvale, CA, Fujitsu Software Corporation).

Associated terminology: C and C++, Java, XML.

Collaborative commerce

Foundation concepts: ERP, Network.

Definition: Collaborative commerce is the sharing of information between two or more entities to improve the collaborators' service delivery or product development.

Overview

Collaborative commerce has fundamentally existed in one form or another since the beginning of modern manufacturing when, for example, manufacturers would share their schedules with a supplier. However, with the advent of more sophisticated technologies such as networking, CAD/CAM design applications, virtual reality models, materials requirements planning systems, ERP systems, and messaging technologies, collaborative commerce has evolved dramatically to the point where corporations link aspects of their value chains together into a value system. The ability for companies to share data across corporate boundaries has been facilitated by the adoption of common base technologies such as XML and TCP/IP protocols. These protocols allow information generated by ERP systems (or similar systems) to be sent across a network efficiently and effectively.

An example of collaborative commerce occurs when company A (e.g., a car manufacturer) sends to company B (a tier-1 component manufacturer) a data file (in XML) that was generated by their ERP system's "planning and forecasting module," which was itself generated in the ERP module that models customer demand. Upon receipt by company B the information is routed into their ERP system and their "planning and forecasting module" re-aligns company B's production levels, inventory levels, human resource requirements, and capital requirements, and notifies their tier-2 supplier of the changes. The majority of these changes and messages can occur without any human intervention at all if the process models are designed to allow such an event to occur.

The advent of internet technologies such as XML has allowed the development of networks that allow many companies to interconnect and share data in large quantities effectively and efficiently. The data-sharing concept that underpins collaborative commerce has developed the concept of a *Data synchronization hub*. Such a hub allows companies to place data pertaining to items in an accepted standard format such as EAN.UUC. The data is then placed in a "data pool" (a database where many companies place their data). This data pool is usually administered by a third party (but for large companies this can be done in-house) and they send information pertaining to the dataset to the global synchronization network (GSN) provider, which holds the information and the location of each item's *Data pool* (q.v.). Customers or other companies then search the GSN registry through their chosen data pool provider and notify the data pool provider of the items they wish to subscribe to. The GSN provider then uses a "synchronization engine" automatically and continuously to synchronize the data between the two companies. For example, a

component's manufacturer places its product specifications in a data pool, then, if they change the specification of a component (perhaps the strength of a washer), all parties who have subscribed to the data pool will be notified of this change.

Business value proposition

Collaborative commerce allows organizations to streamline their operational costs by integrating aspects of their value chain with aspects of partner companies' value chains. Collaborative commerce can be undertaken in any area of joint interest, be it supply chain, logistics, manufacturing, market research, or purchasing. The aim is to leverage technology to bring down costs and speed the delivery of information that is critical to making shared decisions, thus reducing cycle times.

Summary of positive issues

Open standards, protocols, and technologies have evolved to facilitate the entry of companies into collaborative commerce relationships. Collaborative commerce enables companies to perform a variety of strategic undertakings that range from just-in-time inventory models and lean manufacturing to collaborative resource planning.

Summary of potentially negative issues

The large companies leading the push to collaborative commerce may force early adoption of technology upon less technically sophisticated companies, making them absorb the learning and technology costs that accompany early adoption.

Not entering into a collaborative commerce operational model may not be an option in certain industries or sectors in order for a company to remain competitive.

As technologies continue to change, all collaborators utilizing a particular technology will have to assess the impact of those changes and determine their technology-migration policy collectively.

Reference

- G. Norris, J. Hurley, K. Hartley, J. Dunleavy, and J. Balls (2001). *e-Business and ERP Transforming the Enterprise* (New York, John Wiley and Sons).

Associated terminology: XML, Client–server, Internet.

Compiler

Foundation concepts: Programming language, Operating system.

Definition: An application that automatically translates instructions written in a human-understandable language into the computer's own internal electronic representation, so that they become a runnable application.

Overview

Computers and humans do not work in the same way. For a computer, every single step has to be spelled out in explicit, perfect detail. Even the seemingly trivial task of typing the word "hello" so that it appears on a monitor, a task that a human could perform without any further explanation, requires literally thousands of microscopically detailed instructions when carried out by a computer.

In the early days of computing, programmers had to work out for themselves every single detailed step of every task to be performed, and laboriously enter those instructions into the computer. Even the task of adding 10% tax to a list of numbers and printing the results would require a program dozens of pages long. Because so much detail was required from fallible humans, errors were very common, and all programming was a very expensive proposition, requiring highly trained expert technicians working for long periods.

Soon it became obvious that certain subtasks were common to nearly all programs: reading and printing data, calculating square roots, and sorting files are

almost universal requirements. Programmers wrote partial programs for just these subtasks, and fed them into the computer in advance. When a new program was required, only the tasks unique to the new problem had to be programmed, and the computer could be instructed to take a selection of pre-entered partial programs, and *Compile* them together into a single application.

After that first step, there was a flood of development that still continues today. More and more of the tasks of programming that were tedious and error-prone were automated. Gradually it became possible to give the computer instructions in a language that seems almost like English, and allows programmers to express their designs in clear and abstract ways that other programmers can easily understand and assist with. The computer itself accepts these high-level descriptions from programmers, and translates them into the thousands of exquisitely detailed instructions that the programmers used to produce.

A compiler is a computer application that automatically performs this translation from a clear, high-level, *human*-understandable language into the arcane and detailed internal "language" of computer instructions. A compiler is one of the largest, most complex, and most reliable pieces of software on any computer. Using a compiler is effectively making the computer program itself; in the 1950s, programming with a compiler was called *automatic programming*. The human-written program that is the input to a compiler is known as *Source code*; the computer-generated program that comes out is known as *Object code* or *Executable code*.

Business value proposition

The use of compilers allows programmers to be more efficient and effective with their time. As programming is a labor-intensive process it is vital that the programmers and the IT organization do not spend their resources "reinventing the wheel," hence the use of library functions such as "sort" or "merge" embedded within the compiler.

The history of computing includes a large amount of time and resources devoted to the development of "better" programming languages and this has resulted in literally thousands of programming languages and variants being produced since the 1950s, each of which had either an associated compiler or *an interpreter* e.g., Cobol, Fujitsu Cobol, MS-Cobol, Micro Focus Cobol, Dev-C/C++, Algol 60, Algol 68, Fortran IV, Fortran 77, LCC-Win32, Perl, Pascal, Modular, Java, C, C++, Basic, and Visual Basic (VB). The main reason for the existence of all these variants is the fact that there is not just one perfect programming language for all situations and problems, just like there is no one means of transportation for all situations (motorbike, car, van, and truck all suit different needs). The US government attempted to develop one special programming language named Ada that would encompass all its needs and minimize the need for government software developers and contractors to learn and develop code in multiple languages. However, this "one-size-fits-all approach" failed as more and more "special exemptions" were made so that developers could build code for different types of applications. For example, while Ada code may be suitable for payroll systems and accounting systems, it would not necessarily be suitable for code to be embedded in a missile's navigation system. A second factor adding to the complexity of the situation is the nature of compilers: they take the programming language as written at the human level and manipulate it such that it can operate upon the hardware platform (the computer). Unfortunately, until recently a different compiler was needed for each platform, in that a Cobol compiler for an IBM platform would not work for a Fujitsu platform, which resulted in variants

being produced for each machine and frequently the language designers would not stick strictly to the ANSI definitions of the "standard" language, thus causing more variance.

In order to resolve this situation IT organizations within commercial settings have focused on the set of languages and their compilers that are portable or interoperable, that is they will work on any platform. This movement started with the marriage of *Unix* and *C*, which are often *open source* in nature, enabling the programmers to understand the constraints and workings of the systems in which they were programming. By the late 1980s C had become a favored and common development language and evolved into C++, an *object-oriented programming* (OOP) environment, which was more powerful than C yet still portable across platforms. While C++ is the predominant choice of development language for the majority of programmers, other compiled languages that are heavily intertwined with the platform upon which they operate are still widely used, e.g., Visual Basic and J++ are predominantly development languages for interfacing with Microsoft applications and for developing web interfaces.

Summary of positive issues

Compilers shorten the development time for programmers to write programs. The library features of compilers save programmers from needlessly duplicating previous standard functions every time they write a program. Open-source platform-independent compilers allow developers to write code that will run upon a variety of devices.

Summary of potentially negative issues

There exist millions of lines of code written in thousands of programming languages and their variants, few of which will compile and run on any other systems. Compilers are frequently proprietary in nature, being closely coupled to the hardware and operating systems of the computer being used.

References

- R. Wexelblatt (1981). *History of Programming Languages* (New York, Elsevier).
- T. Bergin and R. Gibson (1996). *History of Programming Languages – II* (New York, ACM Press–Addison-Wesley, 1996.

Associated terminology: C and C++, Java, Cobol.

Complexity

Foundation concept: Algorithm.

Definition: Complexity is a measure of the relationship between the amount of data to be processed and the time required to perform that processing. It is not a simple matter.

Overview

In computing, *Complexity* is a technical term that describes the fundamental speed of an algorithm. It is not directly connected with the common meaning of the word, "how complicated something is."

Take for example a simple searching application. It takes as its input a plain text file, perhaps a corporate report, perhaps a novel, it doesn't matter. The application's job is to search the entire file to see whether a particular word or sequence of symbols appears anywhere within it, a common task. Obviously, the longer the file is, the longer the search will take. We could perform a simple experiment and work out the relationship. If we make the application search a 10 MB file, and find that it takes 1 second to do so, we could reasonably conclude that it would take 2 seconds to search a 20 MB file, and 10 seconds to search a 100 MB file.

People naturally expect a linear relationship between the size of a problem and the time required to solve it. If the data size is

doubled, the time required is also doubled; if the size is multiplied by 100, then the time required is also multiplied by 100. This is so much a part of normal life that it is usually just taken for granted. Driving 200 miles takes twice as long as driving 100 miles; reading two reports takes twice as long as reading one report.

Interestingly, the relationship between data size and processing time is often very far from linear, and the assumption of linearity can lead to serious miscalculations. One simple example is "long multiplication," as learned in elementary school. To multiply together a pair of two-digit numbers, four individual simple multiplications are required (to work out 63×75, one calculates 6×7, 6×5, 3×7, and 3×5, then with adding, carrying, and luck, one produces the correct answer). It took me 12 seconds to multiply together those two-digit numbers. With the assumption of linearity, I would expect it to take me 24 seconds to multiply together a pair of four-digit numbers, 120 seconds to multiply together a pair of twenty-digit numbers, and so on.

However, such a prediction is completely invalid. Closer analysis reveals that multiplying together a pair of twenty-digit numbers requires 400 individual simple multiplications, not 40, so it would take me 1200 seconds, not 120. The amount of work required grows with the square of the data size. Different problems (and, sometimes, different solutions to the same problem) produce a different relationship between amount of data and required processing time. Linearity is possible, but must not be assumed.

Non-linear relationships between the amount of data and the time required to process it are very common in computing. A professional programmer must be aware of this, and must know how to perform a basic complexity analysis for an algorithm. This is a major component of a computing curriculum, and it is a very complex subject.

The magnitude of the problem is simple to illustrate. Consider the testing phase of a new software application. It is tested on a data set of 1000 records, and found to take just 1 second to perform its task. A naive developer would assume that processing 10 000 records would take 10 seconds; 1 000 000 records would take 1000 seconds (17 minutes) and so on. The table below shows what the true completion times would be for some common algorithm complexities.

Data size	Linear	Quadratic	Cubic	Logarithmic
1000	1 s	1 s	1 s	1.00 s
2000	2 s	4 s	8 s	1.10 s
3000	3 s	9 s	27 s	1.17 s
4000	4 s	16 s	1 min	1.20 s
5000	5 s	25 s	2 min	1.24 s
10 000	10 s	100 s	17 min	1.31 s
100 000	100 s	$2\frac{1}{2}$ hours	12 days	1.70 s
1 000 000	17 min	12 days	31 years	2.00 s

This is a dramatic illustration, but not a dramatized one. After finding that a new application works properly, and seeing that it takes 1 second to process 1000 data items, many programmers would not bother to test it with 1 000 000 data items, especially after deducing that the test would take 17 minutes. However, if the algorithm turned out to be cubic, it would in fact take 31 years, not 17 minutes, and large-scale customers would be justifiably angry. As the table shows, non-linear complexities can also work the other way, giving unexpectedly fast responses.

Business value proposition

Failure to understand the complexity of a problem makes it impossible to estimate even roughly how long it will take to solve a problem. This in turn makes it impossible to select the appropriate solution for

any given circumstances. Examples of such problems include the algorithms used to search and compare databases. If a company were to develop a new product, say a search engine for the internet, the algorithmic complexity of the approach taken needs to be considered by the organization's programming team. This complexity would include the way the data is structured (the *data structure*) in the *database*, the way the data is placed on the *server* configuration, and the search *algorithm* that operates over the data. In order for the new product to be successful, it is thus imperative that all of the technical complexity issues be considered and that these issues are well understood and resolved such that *platform* and *software* purchases such as the *database* upon which the data will reside can be made in an informed manner. For example, a poorly selected database environment may lead to the programmer using a sub-optimal data structure to store the data, which forces the selection of a sub-optimal search algorithm to be used and leads to a sub-optimal product being produced as a result. Thus, complexity can be directly tied to the value proposition of products as well as the total cost of ownership and return-on-investment decisions associated with a product.

Summary of positive issues

The mathematics of computational complexity, while not easy, is formally developed within the discipline of computation and computer science. The algorithmic complexities of many well-known functions are documented and understood by computer scientists.

Summary of potentially negative issues

Failure to determine the complexity of an algorithm can easily make a software product completely worthless. Not understanding the whole picture of product complexity can be fatal. Algorithmic complexity is constrained and determined by many factors, including budget, hardware, legacy software, and operating system constraints.

References

- D. Knuth (1975). *Fundamental Algorithms* (New York, Addison-Wesley).
- M. Hofi (1975). *Analysis of Algorithms* (Oxford, Oxford University Press).

Associated terminology: Algorithm, Database, Computability.

Compression

Foundation concepts: Bit, Binary, Digital.
Definition: Reducing the size of a data set (the amount of storage it requires) without losing any of the data.

Overview

Compression and *Encryption* are very similar operations. Both convert data into an alternative form, from which the original may be fully recovered later. With encryption, the aim is to make the recovery impossible for unauthorized third parties. With compression, the aim is to make the alternative form as small as possible.

Data in its *Raw* form, as it is collected, unprocessed, tends to contain a lot of redundancy. That is, with some analysis it would be possible to represent exactly the same information in a much less bulky form. Human speech is a clear example. To record one minute of speech faithfully, so that a play-back is indistinguishable from the original to the human ear, requires about 5 000 000 bytes of data. If the only real information in that speech is the words that were spoken, it will amount to not much more than 100 words, or a total of 500–1000 characters, which require one byte each. The same information can be recorded in one two-thousandth of the amount of data. It has lost all the sounds of human speech, but retained all of the useful information. That is one form of data

compression: extracting the true information from the surrounding raw data.

Another form is illustrated by the representation of graphical images in digital form. The most common representation for such images uses a triplet of small numbers (in the range 0–255) for each pixel; the first indicates the degree of redness in the pixel, the second greenness, and the third blueness. So a bright red pixel would be represented by (255,0,0), a greenish-blue one by (0,120,240), a black one by (0,0,0), and a white one by (255,255,255). These numbers are simply stored as they are, in top-to-bottom, left-to-right order, to make an uncompressed image file. A number in the range 0–255 requires exactly one byte of storage (that is why that particular range is used), so a small picture of size just 200×200 pixels would occupy 120 000 bytes uncompressed.

If it is observed that a particular picture involves just four different colors, each of those colors may be given its own binary encoding, perhaps 00 for white, 01 for red, 10 for blue, and 11 for green. Then a sequence of four neighboring pixels can be written in a single 8-bit byte: 00000110 for white, white, red, blue. That results in a total size of 10 000 bytes for the 200×200 image, compressing it to just over 8% of its raw size. Recoding is the simplest form of compression.

If it is further observed that, of the four colors that appear in the image, white is extremely common (perhaps 90% of all pixels are white), red is next (at 8% of all pixels), and blue and green are quite rare (at 1% each), a variable-length encoding will produce additional savings. In this example, a one-bit code, 0, could be used for white, a two-bit code, 10, for red, and three-bit codes, 110 and 111, for blue and green. The sequence white, white, red, white would be encoded as 00100. For the whole image, we would have a total size of 44 800 bits or 5600 bytes, less than 5% of the original size. This is the basic idea behind *Huffman encoding*, which is the basis of many commercial compression schemes.

Further compression may be achieved by noticing that whole areas of the image are the same color, and encoding a representation of that area together with just one copy of its color. Detecting large areas of a single color is very easy for human observers, but computationally difficult. A simpler version, a sequence of pixels within a single row that are all of the same color, is much easier, and is known as RLE, *Run-length encoding*, a popular compression scheme for some limited applications.

Far greater compression ratios may be achieved by noticing that certain patterns of pixel color recur over and over again, and complex encodings that refer back to previous parts of an image to reconstruct patterns can be devised. This is the basis of *LZW* (Lempel–Ziv–Welch) and *GIF* (CompuServe Graphics Interchange Format) compression. Another variation of this provides the logic behind "*Zip files*." Zip files are most commonly used under Windows, but are also compatible with other systems; they are single files that contain "archives" for whole folders or directories all compressed. Individual files may be accessed and decompressed or replaced either individually or in groups. Zip files are a popular and convenient means of distributing whole data collections and applications.

If perfect reconstruction of the original raw data is not required, then *Lossy* (as opposed to *Lossless*) compression methods may give even greater savings. In simple terms, it may be observed that a large region of an image consists not of exactly the same shade of blue, but of a small range of very similar shades. The whole region could be approximated by a single shade region, and the departure from true might not be apparent to observers. More complex lossy methods can involve discovering mathematical functions that approximate the changes in pixel values within a region to any desired degree of

accuracy. Lossy methods may be very good for images, especially if they are only to be viewed by humans, but are not suitable for critical data, where approximating a customer's name or account balance would not be acceptable.

Since bandwidth limitations are a universal fact of life for the computing community, compression methods have been researched intensively. A great many very successful compression algorithms have already been fully worked out and implemented. A programmer faced with the challenge of data compression would be well advised to perform a careful literature search before reinventing the wheel. Compression is possible only when the original data contains some redundancy; compression works by removing or reducing that redundancy. Once data has been well compressed, further compression will not make it any smaller, and in some cases may produce a slight enlargement.

Business value proposition

The use of compression techniques for the transmission and storage of data provides organizations and individuals with the ability to leverage their infrastructure, incurring a relatively low overhead. A principal benefit of compression applications is the ability to store data at a reduced size; while this incurs overhead during compression and decompression, if the data is being stored for the long term (say 7 years for audit and compliance purposes) the storage savings can be considerable and the work to compress the data may be a one-time expense. Data compression on a user's local computer system also allows considerable savings in storage requirements. A popular use of such technology is for the local archival storage of emails in compressed format; this enables users to read old messages if they wish, after decompressing them, but reduces the local storage requirements. Similar processes can be associated with the storage of images or scanned documents. Scanners can encode their images in a bitmap format, producing extremely large but easy-to-process files, but, for storage after processing, a compressed form of the file should be acceptable. All popular image file formats (except BMP) can provide a great deal of compression.

A further aspect of compression that is extremely useful for organizations is the ability to send compressed files over networks, in essence increasing the availability of bandwidth for other tasks.

Network managers can specify processes and enforce policies for the application of compression methods pertaining to their network and its users. They can also ensure that applications whose main focus is the manipulation and storage of data files that are large and require significant resources are optimized. These applications include document-management systems that facilitate the use, linkage, version archiving, and long-term storage of heterogeneous file types (image files, text files, web files, XML files, audio files, video, etc.)

Summary of positive issues

Compression allows large files to be stored and transmitted in a smaller size than their normal format. Decompression allows files to be restored to their original format. There are many well-researched and well-supported compression algorithms, tools, and packages available for the purpose of compression. Compression techniques are built into many software applications and automatically generate compressed files for storage. Most files can be significantly compressed with lossless methods if exact reconstruction of the original is needed, and even further compressed with lossy methods if a good approximation to the original is sufficient.

Summary of potentially negative issues

Compression and decompression impose a resource overhead on the system when the operations are performed. If compressed

files are to be transmitted to other users, the recipient must have the decompression software in order to read them. Repeated compression and decompression using high-ratio lossy methods results in *Generation loss*, in which the data is continuously degraded by each operation.

Reference

- K. Savood (2000). *Introduction to Data Compression*, Morgan Kaufmann Series in Multimedia and Information Systems (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Audio, Video, Information lifecycle management.

Computability

Foundation concept: Algorithm.
Definition: What tasks are really impossible to perform with a computer system?

Overview

The concept "impossible" is usually confused with the concepts "I don't know how to do that" and "that isn't within our technological grasp." It is often said that it is impossible for anything to travel faster than light, but the truth is simply that our current theories of physics do not allow anything to accelerate beyond the speed of light. We do not really know that these theories are correct, and human understanding of physics has been completely turned on its head any number of times in the past. People may say that it is impossible to fly to the moon for less than $300; we can't do it now, but we have no idea what technological advances the future may hold.

Similarly in programming, there are many things that we just don't know how to do. True artificial intelligence, a computer mimicking a human well enough that an impartial blind observer can not tell that it is not really a human, has not been achieved, and nobody knows how to do it, but it is not known to be impossible. In fact, we know that it *is* possible, we just don't know how. The ability might just be round the corner, or we might never find it out.

In the physical world, most things classified as impossible may well not be. With the speed-of-light example, it may be that new discoveries in physics will completely change our beliefs; that does not make impossible things become possible, it simply makes us realize that we were wrong in the first place.

In computer technology, things called impossible are *usually* merely beyond our reach. An expert may say that 2048-bit RSA encryption is safe, meaning that, with currently foreseeable technology, encrypted messages will remain unreadable. But nobody really knows how technology will develop in the future. Totally new, currently unimagined, designs for computers could render all these assumptions invalid. The idea of a *Quantum computer* is currently no more than science fiction, but, if a practical one were to be constructed, it would have exactly this effect.

Additionally, there is the possibility of new developments in the theory of computation. Continuing with the same example, one of the great unknowns is the "*P equals NP*" question, which many theoreticians have devoted their entire careers to trying (and failing) to answer. Nobody knows whether P equals NP or not, and currently it is a very dusty academic subject. But if the answer is ever found, the results will be striking. If it turns out that P does not equal NP, then RSA encryption really is safe. If it turns out that P equals NP, then immediately, without any further work, literally thousands of intractable problems suddenly become easy; cracking RSA encryption (and a lot of other things) instantly becomes viable. What was called impossible is now possible.

To make matters more complex, theoretical computing does know of some things

that are genuinely impossible. Things that, no matter what new technological leaps or theoretical insights occur, must remain completely impossible; problems that if solved would result in a contradiction in logic itself. The study of computability, that is, possibility and impossibility in computer processes, is deeply complex and very easy to misunderstand. As a general rule, only those with advanced degrees in computer science or engineering are even aware that the subject exists, and far fewer have any expertise in the matter.

Business value proposition

Any attempt to do the impossible is likely to be a financial disaster. A belief that something is impossible when it isn't can result in a technological disaster. Whenever a programmer claims that something is impossible, it is important to find out what they really mean by that, and whether they really have a valid basis for the claim. For example, the ability of a program to check itself for correctness is theoretically impossible and this impossibility derives from *Turing's thesis*, thus it is important that, should claims such as "self-checking software" be made by vendors, this should be thoroughly examined and understood before a program is used in what could be a critical process, such as one involving a threat to human life.

Summary of positive issues

Computability is a theoretically sound field of study and significant amounts of research literature exist to help computer scientists and professionals examine the underlying mathematical and logical basis of their programs and specifications.

Summary of potentially negative issues

Computability theory is an extremely specialized and complex field of endeavor, which is primarily studied by researchers and academics. Programmers without advanced course work in computation may have only a limited understanding of and exposure to the discipline and its impact upon their work.

References

- J. Hopcroft and J. Ullman (1979). *Automata Theory, Languages, and Computation* (New York, Addison-Wesley).
- J. Brady (1977). *The Theory of Computer Science* (London, Chapman and Hall).
- A. Turing (1936). "On computable numbers with an application to the *Entscheidungsproblem*," *Proceedings of the London Mathematical Society*, **2** (42), 230–265.

Associated terminology: Algorithm, Complexity.

Computer

Definition: A calculating and data-manipulating device capable of storing its own program of instructions and following those instructions without intervention.

Overview

Everybody who is reading this will already have a fairly good idea of what a computer is. Instead of discussing unnecessary technical details, this article will present the major varieties of computer currently available.

Palmtop, **PDA** (portable digital assistant). These are computers so small that they can be held in the palm of one hand. Generally they have much lower processing power than any other kind of computer (because the very restrictive space requirements leave very little room for powerful batteries) and very restricted storage capacity. Some, but certainly not all, are restricted to running a small range of specially reduced software applications. They have small displays and either miniaturized keyboards or none at all (using a stylus instead), so they can be quite unsuitable for normal office work.

Notebook, laptop, portable. These terms have grown to mean almost exactly the same thing. Originally, a portable computer was one that could be carried around, but doing so was not necessarily a pleasant experience; such computers are generally not made any more. When shades of meaning are intended, a notebook is usually an extra-small laptop.

Desktop. A general term that now covers nearly all computers in common use. Any computer that can fit on a normal desk and still leave enough of the desktop available for other uses.

Server. The term "server" properly applies to a software configuration: a network application that provides some service to a set of client systems on demand. The word is often used to describe a desktop-style computer that is at the higher end of the performance range, and so might reasonably be used for running server applications. The term is not strictly defined, and servers range from a personal computer that runs applications that do not require especially high performance to clusters of high-performance computers that run enterprise-class systems such as ERPs.

Workstation. The counterpart to "server." The term is not strictly defined, but is generally taken to mean a desktop-style computer intended for use by one person at a time. Workstations are not at the top of the performance range.

Thin client. A very-low-power workstation, often one that is used only for display purposes, all the real work being done by a server that receives commands from, and transmits responses to, the thin client over a network connection. A thin client could be a reasonable means to provide restricted public access, perhaps including just web-browser capabilities.

Diskless workstation. A variety of thin client, which has no disks or other permanent storage media of its own, and so is totally reliant upon a server for all operations. Since disks are a significant component in the cost of a computer, diskless workstations can be financially very attractive. The performance of a diskless workstation can be very poor, because it is strictly bounded by available network bandwidth.

Microcomputer, minicomputer, mainframe, super-computer. These are vaguely defined points on the spectrum of overall processing power for computers. A microcomputer is normally taken to be any computer whose entire CPU is on a single integrated-circuit chip. This covers all desktop computers now, and the term has fallen out of use. Originally microcomputers were thought of as essentially toy computers that were not suitable for any serious work. Minicomputers were computers that were really suitable for use by only one user (or a very few concurrent users), but were certainly not just toys. This term has also fallen out of use, since the realms of microcomputers and minicomputers have completely merged to create the desktop computer. A mainframe is a high-power, high-price computer, usually very large and having a staffing and facilities requirement to keep it running (e.g., air-conditioned rooms, automated non-destructive fire extinguishers, and heavy-duty uninterruptible power supplies). Mainframes provide much faster processing and storage access, and much higher input and output bandwidth than other computers, typically being able to support thousands of concurrent users. A super-computer is simply a very fast mainframe, often supporting fewer users and having a lower access bandwidth than a typical mainframe, but providing even higher computing speed.

Embedded system. Not all computers are clearly identifiable objects intended for direct use by people. Often computers are built into other devices as part of the control or monitoring system. Modern cars, microwave ovens, and even washing machines are computer-controlled. An embedded system is a computer that is

built into something else, not running general-purpose applications, but directly interfaced with the system that it helps to control.

Business value proposition

Computers are not classified just by the above terms. The question "should we get a Windows, a Linux, or a Unix computer?" is frequently asked. The important question to consider is "what does the computer need to be able to do?" In order to answer this, it is important to understand the required performance characteristics for the processes that the system is to support. However, note that we now talk of "system" rather than "computer" and link the computer itself to the operating system and the applications on top of that, coupled with the databases, and linked through networks.

The first layer of the "system" above the hardware of the computer itself is the operating system. A similar assessment is required for the choice of operating system(s) to be deployed. An investigation of the applications, databases, and other systems planned (or in place and to be integrated into it) will need to be undertaken. For example, some database systems will not run on specific popular operating systems, while some ERP systems will run on any operating system and thus an assessment of the inter-operability of the overall systems architecture needs to be undertaken.

The technical assessment of inter-component compatibility needs to consider the loads that are to be placed upon the system, e.g., is the system intended to support a factory with relatively low throughput or a factory that supports high-volume mass production? How many users will be using the system at peak load times, what are the end-to-end response times that are required of the system, etc? The technical issues and capacity issues will then enable a total cost-of-ownership assessment to be undertaken.

Summary of positive issues

The wide variety of computer specifications enables systems designers to create and configure systems to meet their individual systems specifications.

Summary of potentially negative issues

Technology has a distinct "half life" and needs to be assessed continually regarding its suitability to support the organization and its processes. Technology is not universally compatible.

Reference

- M. Campbell-Kelly and W. Aspray (2004). *Computer: A History of the Information Machine* (New York, Basic Books).

Associated terminology: Hardware, Operating systems, Client–server, ERP.

Computer-based training (CBT)

Definition: Computer-based training is the use of computers and related technologies to provide an instructional learning environment through which human subjects can interactively undertake an educational experience.

Overview

Computer-based training (CBT) involves the use of computers, software, and network technologies to provide an educational platform. Various system configurations are utilized, including internet delivery, CD-ROM-based delivery, and LAN delivery over a corporate intranet. The technologies used for content include video- and audio-delivery formats, video-assisted interactive simulation training, interactive quizzes and exams, and the delivery of case studies combined with exams.

Business value proposition

CBT is a popular tool used by organizations wishing to provide an online educational experience. The tools can be placed online,

allowing employees to undertake training in accordance with their schedules. Training can be repeated until satisfactory performance levels have been achieved. Corporations can monitor employee performance and compliance levels through CBT. CBT allows flexible training strategies and testing methods to be developed; certification systems frequently use adaptive testing in which the questions asked in a test get harder or easier depending upon a student's results up to that point in the test. The CBT model provides a low-cost means to provide flexible training for the workforce. CBT is of particular benefit to companies that experience large turnover rates in training their workforce, since it allows for repetitive training in standard processes at a low cost.

Summary of positive issues

The principal advantages of CBT are the lowered delivery price for common courses, the ability to provide an educational experience for employees that can be undertaken at the discretion of the student, and the ability to provide courses that can be taken repeatedly by students until the level of material comprehension is acceptable. The CBT method allows students to pace themselves through the materials or to use the system to undertake training as it fits their schedules.

Summary of potentially negative issues

Development of CBT systems can be expensive and time-consuming. CBT systems still require human management and students need to have a mechanism through which questions may be addressed. Specialized courses or topics that are time-sensitive in nature might not be applicable or cost-effective.

Reference

- R. Clark and R. Mayer (2002). *e-Learning and the Science of Instruction: Proven Guidelines for Consumers and Designers of Multimedia Learning* (San Francisco, CA, Pfeiffer).

Associated terminology: CD-ROM, LAN, Internet.

Computer Fraud and Abuse Act, 1986

Foundation concept: Security.
Definition: Federal law forbidding unauthorized access to data stored on computers owned by the federal government and financial institutions, or access to further the commission of fraud.

Overview

The act forbids intentional access to a computer that either is unauthorized or exceeds authorization (covering a user who is authorized to use a computer and has a valid account, but uses it to access data that he or she is not authorized to access). However, the act covers only accesses to data that are covered by national security concerns, accesses to data on computers owned by the federal government, and cases in which financial information is accessed on a computer owned by a financial institution.

The act also forbids unauthorized access to any computer when the intent is fraud and the access furthers that fraud, except when the only fraud is the unauthorized use of the computer and the value of that use is less than $5000 in a year.

Section 5 of the act is of more general interest, since it forbids "the transmission of a program . . . or command" that "causes damage without authorization" to a "protected computer." This is a clear description of releasing a malicious worm, virus, or Trojan horse. However, to be covered, the damage caused must add up to at least $5000, result in some injury or threat to public safety, or involve a government computer.

Section 6 of the act forbids the unauthorized dissemination of passwords, if the

intent is to defraud, and it affects interstate or foreign commerce or a government computer.

Business value proposition

The legislation primarily covers intentional misuse or unauthorized access to federal (US) computers, equipment, and resources. The act also covers general misuse of computing resources (such as those at corporations) resulting in "damage." The legislation acts as a deterrent to misuse and provides the judiciary with the ability to impose penalties upon those found guilty.

Summary of positive issues

The act forbids intentional unauthorized access to a computer system and access that exceeds authorization. The act covers the misuse of access rights to inflict damage upon other systems, including medical and government systems, damage to individuals in the form of physical injury, and threats to public safety. Financial damage from misuse is set at a lower bound of $5000.

References

- 18 USC 1030; Public Law 99–474.
- http://www.usdoj.gov/criminal/cybercrime/1030NEW.htm.

Associated terminology: Virus, Trojan horse, Law cross-reference.

Computer Misuse Act of 1990

Foundation concept: Security.
Definition: The law of the UK which covers unauthorized computer use.

Overview

Section 1 of the act clearly states that a person is guilty of an offence if they cause a computer to perform any function with the intent of accessing any program or data on that computer, if they know that that access is unauthorized. The act explicitly states that the intent does not have to be to access any particular program or data, or even to access any particular computer, so simply releasing a virus or Trojan horse on the world at large is covered if it ever works. The offense can earn up to five years' imprisonment.

Section 3 of the act adds that it is an offence to do "any act which causes an unauthorized modification of the contents of any computer." Section 17 clarifies the position, stating that it is an offence just to cause a program to be run without authorization.

Spyware (q.v.) is, in its most common usage, software designed to provide information from a computer, without its owner's consent or knowledge, to a third party. This is fully covered by Section 1 of the act, which forbids simply accessing data without authorization, regardless of whether any fraud is committed or any harm is done, or even whether any files are modified.

Section 3 covers unauthorized modifications of the desktop setup, changing a web-browser's default "home page," and introducing viruses and worms into the system.

Business value proposition

The legislation primarily covers the intentional misuse of any computer within the UK and its jurisdiction, either directly or through remote means such as releasing a Trojan horse. The act also covers the unauthorized modification of a database or the contents of a computer. The act also is powerful in that it covers the situation when a user has accessed data to which they are not authorized.

The act provides a strong deterrent against the misuse of computing resources at corporate level by malicious or even over-inquisitive individuals.

Summary of positive issues

It is a wide-ranging act that protects computer systems from misuse through attack by harmful software and from users who access information to which they are not authorized. It covers all computers.

Reference

- "Computer Misuse Act 1990 Elizabeth II. Chapter 18," The Stationery Office Limited, 123 Kingsway, London WC2B 6PQ, UK.

Associated terminology: Law cross-reference.

Computer Security Act of 1987

Foundation concept: Security.
Definition: The act aims to establish standards and procedures "that improve the security and privacy of sensitive information in federal computer systems."

Overview

The Computer Security Act of 1987 was enacted "to assign the National Bureau of Standards responsibility for developing standards and guidelines for Federal computer systems, including responsibility for developing standards and guidelines needed to assure the cost-effective security and privacy of sensitive information in Federal computer systems."

The act requires the NBS to produce guidelines for a wide range of areas, including technical, managerial, physical, and administrative areas, aiming to prevent unauthorized access to federal data systems. The act also calls for training to be provided in accordance with the goal of proving data security and assessing system vulnerability across agencies. The implementation and status of the act are monitored by a Computer System Security and Privacy Advisory Board within the Department of Commerce.

Reference

- Computer Security Act of 1987, Public Law 100–235 (H.R. 145).

Associated terminology: Law cross-reference.

Connectivity standard

Foundation concepts: Network, Client, Server.
Definition: A connectivity standard is any protocol used by two or more devices to communicate.

Overview

The term *Connectivity standard* is used widely to denote a variety of standards or protocols that are utilized to enable two or more devices to communicate and share data.

A set of standards is proposed and administered through authorities and bodies such as ANSI, which is responsible for X.12, and the World Wide Web Consortium, which is responsible for XML. These standards are aimed at worldwide cross-industry coverage and acceptance.

A second set of standards is proposed and administered within specific industries to cover special regulatory, technical, or data requirements. For example, the National Committee for Clinical Laboratory Standards developed a "vendor-neutral standard" known as *POCT1-A* that permits medical point-of-contact devices from different manufacturers to communicate with laboratory and hospital information systems (e.g., hand-held devices that perform analysis on blood for levels of electrolytes, proteins, or glucose).

A third set of standards is proprietary and developed by organizations that are associated with a product: data transmission over networks was subject to vendor-devised standards prior to the adoption of TCP/IP. Proprietary standards shared between two or more entities are also developed to solve a particular technical,

business, or data need but are not usually considered "standards" in the universal sense of those supported by bodies such as ISO and ANSI. However, should the technology developed by a vendor become popular, indiscernible, or even forced upon customers, such protocols may be known as "industry standards," but should still be considered proprietary in nature.

There is a fourth set, termed "open standards," for which the specifications and protocols are explicitly available to anyone who wishes to adopt them or modify them.

Business value proposition

A wide variety of connectivity standards is available to facilitate connection between hardware, software, and network devices, through both wired and wireless media. The use of widely adopted standards allows organizations to connect with a wide spectrum of users who have also adopted that standard. International and national standards are typically adopted and used by technology vendors to enable their products to connect with other products and are developed to maintain compatibility between vendors. Open standards allow organizations to amend and modify technology protocols to meet their particular requirements.

Summary of positive issues

There exists a wide range of connectivity standards that allow entities to communicate in a variety of ways and incur the overhead that is appropriate for their requirements. International, national, and industry-specific standards tend to have wide support and established consulting resources are available to establish, maintain, and support them.

Summary of potentially negative issues

The adoption of proprietary connectivity standards imposes a degree of dependency upon the vendor and it may be necessary to upgrade or change at the vendor's pace rather than independently. Such standards, not developed by authorizing bodies, might not become adopted by a wide user base and could lead to problems in connecting with other entities that adopted what turned out to be the "industry best practice" or de facto standard.

Some vendors do not always support all the connectivity standards, and a compromise may be required in order to enable communications. For example, the development of a middleware solution or simply the adoption of a second choice, which is a more supported standard, may be required.

Associated terminology: Protocol, Client–server, W3C, IEEE.

Cookies

Foundation concept: e-Commerce.

Definition: A cookie is a small packet of information created by a web server and stored on a web-browsing computer without the knowledge of the computer's owner. It contains information about recent transactions with the server, and is used to identify returning browsers and provide some context for their visit.

Overview

Browsing the web is a *Stateless* activity. This means that, when a browser visits a web site, an internet connection is made between the browser and the server, the browser transmits a request for a document or page of information, the server transmits the requested information, and the connection is terminated. If a second document is needed from the same server, a new connection may be required. No information survives from one request to the next. Even if a complex interaction is under way, in which the user is completing a multipart *Form* to supply essential data for a transaction, each stage is completely disconnected. A web server works by listening for anonymous requests for data, servicing

those requests, and immediately forgetting them.

This would make reliable commercial transactions impossible. If the browser's order for a product, the server's response with the price, and the browser's provision of credit-card data and shipping address can not be kept together, the system is not going to work. Cookies are one possible solution to the problem.

A web server can create a cookie that contains within it essential details of a transaction, and send it to the web browser along with the information that is to be viewed. The browser will store the cookie, possibly permanently, on disk. Every time a connection is made to a server, the cookies previously sent by that server are sent back to it. Thus the server can reconstruct earlier parts of the transaction. The server must usually record all cookies that it has sent for validation and other purposes.

Cookies can persist on the browser's computer for an unlimited time, and create a record of the owner's web-browsing history that the owner may well rather not have exist. There is a widespread perception that cookies can extract private information from a computer and secretly transmit it back to a server. There can be no doubt that some insecure systems have allowed cookies to be released to unauthorized third parties, creating a potentially severe violation of privacy and contributing to identity theft. Cookies are not active; they can not "scan" a computer or erase its files. It is technically possible for a virus to be transported by a cookie, but some other means would be required to activate it.

Business value proposition

Bad publicity surrounding cookies generates dissatisfaction with sites that insist on using them. Other solutions to the stateless-server problem may be available and should be considered. However, there is nothing inherently wrong or privacy-eroding about cookies, and when they do provide the best solution, some clear statement of policy may calm users.

The "cookie" debate is centered upon third-party privacy issues, where companies use cookies to develop "one-to-one" marketing campaigns aimed at the individual computer and its user. Primarily a company will contract with a marketing company to develop profiles of its customers in order to develop customized responses and advertising. Typically, the marketing company places cookies on their clients' sites and, since a cookie can be associated with any object (e.g., a banner advertisement, download icon, or web page), the profile for individuals' surfing habits can be developed for that specific company's web site.

Marketing companies also aggregate data by placing cookies on multiple sites and thus, when a customer visits any of these sites, the information pertaining to a particular cookie is not actually sent directly back to the company whose site is being visited but rather the information is sent to the marketing company, allowing aggregated data to be stored. The aim of this practice is to create an overall score for an individual that can then be passed on to a client company summarizing an individual customer's total web usage.

In light of the controversial practice of cookie aggregation, the Internet Engineering Task Force, an influential body through which internet standards and policies are developed, is examining the issue of "third-party" cookie requests and is developing policy positions that may limit or stop such practices being employed without the customers' consent.

Summary of positive issues

Cookies are a simple device that eases and facilitates internet and electronic-commerce transactions. They are universally available on all platforms and permit reliable transactions to take place.

Summary of potentially negative issues

The negative public perception of cookies has grown from the use of cookies by third parties who aggregate the cookie data and profile the users, which can be perceived as an invasion of privacy by computer users or even an unacceptable security risk for many users.

References

- D. Kristol and L. Montulli (2000). "RFC2965: HTTP State Management Mechanism" (proposed standard), http://www.ietf.org/rfc/rfc2965.txt.
- http://www.cookiecentral.com/dsm.htm.

Associated terminology: Web server, Security.

CPU (central processing unit)

Foundation concepts: Hardware, Computer.
Definition: The functional unit in a computer responsible for performing arithmetic and logical operations, carrying out the execution of programmed instructions, and controlling the whole system.

Overview

In the days of mainframe computers, the circuitry that performed the arithmetic and control functions of a computer would usually occupy at least one quite large and impressive looking box, adorned with flashing lights and switches. The memory would occupy another nearby box, and there would be a lot of other large boxes also connected to it, responsible for controlling disk drives, magnetic tapes, printers, and so on. These boxes had to be given names, and were called the *Central processing unit* (or CPU), *Memory* unit, and *I/O* (input and output) unit, respectively. These designations have survived as names for identifiable parts of modern computers, although they are much less significant, and serve more to give students something to memorize and be tested on than for any usefully descriptive purpose.

In a desktop or personal computer, the term CPU is now usually used for the single chip (integrated circuit) that provides the "brains" of the computer, such as a Pentium IV, a G4, or an Alpha, to name but a few, even though it also contains a significant amount of cache memory and I/O control. The term *Microprocessor* is usually used for any CPU that consists of a single integrated circuit.

There are three distinct schools of thought on overall CPU design philosophy. The traditional philosophy holds that CPUs should be designed with as many bells and whistles as possible, able to perform complex operations such as evaluating polynomials and rotating four-dimensional vectors with single instructions. This is known as the *CISC* (Complex Instruction Set Computer) design, and probably reached its acme in the now-discontinued DEC VAX line of processors. An alternate philosophy is that, if the processors are made as simple as possible, limiting their abilities to performing simple arithmetic and logical tests, they will be able to work much more quickly and efficiently. This is known as the *RISC* (Reduced Instruction Set Computer) design, and probably reached its acme in the now-discontinued Inmos Transputers. A third philosophy is that CPUs should not be powerful or complex or fast; a large number of very cheap processors cooperating on one task can do a better job than one very fast powerful processor. This is the *CSP* (Communicating Sequential Processors) design, and is currently re-emerging in IBM's Cell Computers.

Which of the three philosophies is chosen may have little effect on the end user, or even on applications programmers, but, for organizations involved in low-level (operating systems and languages), real-time, or video-game programming, the basic design of the CPUs to be used is a major consideration.

Business value proposition

The choice of CPU for most users is either irrelevant or beyond their control. A purchaser may favor brand X over brand Y, but typically purchasers buy on the basis of perceived macro factors such as speed, memory, and price. Unless they are especially sophisticated they will typically not be able to balance the trade-offs that are occurring even at the macro level, just equating more speed and memory with higher quality and then trading off cost versus quality. The issues surrounding the CPU's ability to function effectively for any particular application and data are rarely considered. It should be noted that CPU manufacturers do market some chips as more suitable for multi-media purposes than others, but the same overall constraints on systems performance apply.

In the future CPU designers may move from one structure to another, say from RISC back to CISC systems, or they may vacillate somewhere in between. The primary impact this will have upon the corporate IT executive will be on the choices that they will have for software–hardware combinations. Should one design approach become technically more advantageous, it does not always follow that software vendors will develop code for that platform, and, as consolidation amongst the mega-vendors continues, this may limit a CIO's technology-purchasing options.

Summary of positive issues

CPUs continue to evolve following Moore's law, which states that the number of transistors on a chip doubles about every two years. CPUs are highly reliable under normal operating conditions. The CPU architectures are fairly well documented and supported.

Summary of potentially negative issues

CPUs are built by a limited number of manufacturers using proprietary designs. As CPU capacity expands, the support systems need to be enhanced to enable the CPU to function.

References

- C. Thimmannagari (2004). *CPU Design: Answers to Frequently Asked Questions* (New York, Springer).
- http://www.intel.com/technology/silicon/mooreslaw/index.htm.

Associated terminology: Cell computing, Bus, Clone.

Cracking

Foundation concepts: Security, Password.

Definition: Revealing encrypted data, passwords, encryption keys, and secret product-activation codes by analysis.

Overview

The idea of "cracking codes" is well known. An analyst pores over encrypted message intercepts, theorizes, analyzes, and guesses until the process is discovered, the code is broken, and future intercepts can be decoded with much less effort. Cracking is a process of intellectual analysis and experimentation. If one is fortunate enough to find a decrypted message, one can not claim to have cracked the code. Cracking in the world of computer security is exactly the same.

Computer systems use hidden secret information for a wide variety of purposes. Apart from the obvious reason of wanting to keep confidential data private, there are more subtle operational reasons. When an authorized user accesses a computer, they need to identify themselves in some way; this is usually done with a non-secret username coupled with a secret password. The computer system must be able to verify that the password entered is correct, so it must have some record of the correct password. Naturally, password files are kept in a strongly encrypted form.

When an expensive software application is sold, the customer must have some way of installing it on their computer, and reinstalling it after something goes wrong; this is easily handled when software packages are distributed on CDs. Since the early 1990s it has been very easy for anyone to make copies of any CD. This causes a significant problem for the software manufacturer, since one legitimate sale may result in hundreds of black-market copies entering into circulation. In the age of high-speed personal internet connections, it is very common to find the entire contents of popular CDs freely downloadable from public web sites. A common solution is for software manufacturers to design their products so that they will work only after a secret identification code has been entered; the product may communicate with headquarters (over the internet) to ensure that no two products have been activated with the same key. Manufacturers must design a system that allows them to generate a very large number of different keys, all of which will be accepted by the software, whilst keeping it impossible for customers to generate those keys for themselves.

There is a major underground industry devoted to cracking all of these applications of software security. Anyone with an internet connection can freely download cracking kits that do a surprisingly good job of discovering any poorly chosen passwords. For most software that requires an activation key, there is software available that will instantly generate new valid ones. Of course, every major government has a department devoted to cracking all of the major encryption algorithms (we may take some comfort from the fact that there is no evidence of success in this direction for any of the modern strong methods).

The distinction between a true crack, software that actively discovers passwords or generates activation keys, and simply publishing a known password or key, is vitally important. Many "cracks" web sites do simply provide a list of known keys, many or all of which have simply been copied from people who legitimately bought the software in question. This is not a crack, and it is a simple matter for a savvy software manufacturer to make each of the published keys invalid. Once a key-generation method has been discovered and made public, there is very little that a manufacturer can do about it without cutting off all of their legitimate customers.

Business value proposition

The "industry" of cracking is an aspect of the software culture that developers and network owners need to be aware of. Understanding the risks associated with password and access-control mechanisms is a vital aspect of controlling intellectual-property ownership and security.

Summary of positive issues

Security and control of passwords and access mechanisms is an aspect of systems development and ownership that is controllable by the use of carefully designed and conscientiously implemented formal processes to manage access, storage, and use of critical information.

Summary of potentially negative issues

Cracking is endemic and a part of the software culture; it is unlikely to go away.

Associated terminology: Hacker, Encryption.

Database

Definition: An organized collection of information.

Overview

A collection of files and records is just raw data. Once it has been organized, rationalized, and formatted so that it can be effectively accessed in a uniform way, enabling general queries to be answered by a simple procedure, it becomes a database.

Rational organization is of course essential for any effective record keeping, but for a database system the demands are far more rigorous. A database system will be expected to be able to answer queries totally automatically. To answer even the simplest of queries, such as "what is the salary of the employee named Q. X. Smith in the advertising department?," mechanically, without any access to intelligence, requires an established format to the data. A human searcher could simply leaf through all the employee records; they would easily be able to pick out the employee's name and their departmental assignment from each form, because everybody knows what proper names look like. It is just common sense; nobody would ever confuse a department name with a social-security number even if they were written in unexpected places on the form.

Of course, computers don't have common sense, and can't be expected to know what makes a credible name for a human. For any such query to be answerable, every single employee record must be formatted uniformly, so that the query application knows exactly where in each record to find the employee's name, department, and salary. There must be no exceptions or slip-ups; if an employee's name is recorded as "Research and Development," there is no way a computer system could react appropriately.

The first step in setting up a database is to decide what different kinds of records it should contain. For example, an organization may have a number of different paper forms used to record basic employee information, but they will all contain essentially the same information, with perhaps a few minor differences. It would make perfect sense to design one kind of employee record that covers all cases, so that they may all be kept together, and searched with single simple queries. On the other hand, the information kept on customers is likely to be significantly different, and the information on supply levels for raw materials even more so.

A formalized description is created (using a *Data definition language*, DDL) for each kind of record that will exist. This description specifies exactly what individual items of data will be present (e.g., first name, last name, social-security number) and their formats and ranges (a string of up to 30 letters, a nine-digit number, etc.). Using this description, a *database management system* (DBMS) is able to set up the very large disk files that will eventually contain all of the real data, in a manner that makes searching simple and fast. All of the data descriptions, which are really data about data, are known as the meta-data, and are stored as part of the database for future reference.

In database terminology, the entire collection of records of a single type is known as a *table* (hence a query may specify a search of the "employee table"); many DBMSs keep each table as an individual very large disk file, but this is not always the case. The records within a table are actually known as *records*, and the individual data items that make up the records (names, numbers, etc.) are known as *fields* or *attributes*. Each field has a *name* (e.g. "social-security number") and a *type* (e.g., "nine-digit number").

Once the tables have been set up, they must be populated with their data. If the organization already has the data recorded

in some electronic form, this will require some simple but specialized programming to perform the data conversion automatically. If the data does not exist in electronic form, then a long period of manual data entry will be required. Paper forms can be automatically scanned, and, although scanners now have very good resolution and high reliability, the written information must be converted into a computer-recognizable digital form. *Optical character recognition* (OCR) technology is not sufficiently advanced, especially for decoding hand-written characters, that it could be relied upon for commercially or legally essential data. The time spent verifying and correcting the results of automatic OCR can be as much as the time taken to enter the data manually.

Once the data is entered, a DBMS application usually provides some default user interface in the form of an interactive *Query language*, the most popular of which is SQL, the *Structured Query Language* (pronounced "Ess Cue Ell"; *Sequel* is a slightly different thing, a complete DBMS software package, rather than just the language). SQL is a standardized language for all database operations; it covers database creation, and record entry and deletion, not just queries, and it can be mastered by nearly all after some technical training, but it can not be reliably used by unskilled untrained employees, so most organizations also require some significant expenditure in software development to produce a safe and reliable user interface. Commercial DBMS software, especially that intended for use on desktop and similar systems, often provides a more intuitive and user-friendly graphical interface, but it still requires a fair amount of training before it can be used effectively.

Nearly all database systems in use today follow the *Relational database* model. That means that some real information is represented not by records in single tables, but by the relationship between data items in different tables. For example, to record the fact that Q. X. Smith sold 75 brown cows to Amalgamated Meats Incorporated for $15 000 on July 4, 1993, there might be an entry in the employees table indicating (amongst other things) that there is an employee named Q. X. Smith with ID number 3636; there might be an entry in the products table indicating that there is a product named Brown Cow with UPC 2135712; there might be an entry in the customers table showing a customer named Amalgamated Meats Incorporated with reference number 25008, and there might be a record in the sales table containing just the six numbers 3636 25008 2135712 75 15000 19990704. This organization saves a lot of space in the data files, and goes a long way toward ensuring data consistency (see *Normalization* for further explanation), but does mean that the procedure for answering simple queries like "What is the total value of sales made by Q. X. Smith?" becomes quite complex.

DBMSs, even light-weight single-user versions for personal computers, are often not stand-alone applications, but a client–server pair. A client provides the front end, interacting with the user and composing correct queries, perhaps even rendering the user's commands into SQL. The command is sent to the *Database server*, a second application, not necessarily on a different computer, for processing. The eventual responses are then reprocessed by the client for display in a suitable format. The use of a database server even when only one computer is involved simplifies design, and is a great aid to scalability and accessibility. It is a simple matter to change the connection to a remote server when conditions demand, but continue to work as before. When a database becomes too large to handle locally, multiple database servers are used, making a *Distributed database*.

Business value proposition

The use of relational databases has become almost universal in standard database file systems. The relational system is at the center of modern applications, and systems such as enterprise resource planning (ERP) systems take full advantage of their capabilities. ERP systems, for example, use a single database to store and manipulate their data. This is an efficient and effective approach to data management since they will, for example, contain a single occurrence of a customer's name, address, ID, etc., and this prevents problems such as data redundancy, where different addresses occur for a customer on two or more databases.

The range and scope of relational database technology have evolved significantly since it was formulated by Ted Codd in 1970. At one end of the spectrum, database technology is available on a user's desktop computer through a small database application; at the other end, large corporate databases operate on specialized high-performance servers. To ensure that the database created at an organization is technically correct and matches the needs of organization, a database administrator (DBA) is employed to oversee operations.

Summary of positive issues

Database theory is highly developed in the academic and practitioner literatures. Human resources to develop, run, and maintain databases are widely available. Certification by vendors is available to ensure the training levels of the employees.

Summary of potentially negative issues

Creation and execution of database systems requires specific technical training, and the technology continues to evolve. Some database systems such as the older hierarchical model have fallen out of favor. Continuous updating of skills may be necessary.

References

- C. Date (2000). *An Introduction to Database Systems* (New York, Addison-Wesley).
- R. Elmasri and S. Navathe (1994). *Fundamentals of Database Systems* (New York, Addison-Wesley).
- J. Bowman, S. Emerson, and M. Darnovsky (2001). *The Practical SQL Handbook* (New York, Addison-Wesley).

Associated terminology: Database administrator, CIO, Distributed database, Application server.

Database administrator (DBA)

Foundation concepts: Database, ERP, CIO.
Definition: A database administrator is an expert IT professional who maintains and optimizes the corporate databases.

Overview

The use of data within a corporation is central to any information system, and thus the organization and use of that data are critical to optimal performance of corporate information systems. To ensure that the data within a company is designed and utilized correctly, companies appoint a professional database expert known as a *Database administrator* (DBA), who reports to the CIO.

The role of the DBA is to act as the authority pertaining to any data used in the corporate information systems. While in small companies the DBA may also be a part of the programming team, in larger organizations the DBA heads a data-administration group. The duties of a DBA include planning, designing, and implementing the databases of the organization as well as enforcing security measures and ensuring operational performance. Specific technical issues for which the DBA is responsible include advising on the selection of

database-related products, e.g., data warehouses, and products that interact with database systems, e.g., ERP systems. The DBA is responsible for the physical database design, including changing existing databases to accommodate modified or new applications. Other duties include the training of employees in database technologies and development of contingency-planning activities with the CIO and chief security officer so that any data corruption or loss can be remedied quickly and effectively.

Business value proposition

The DBA performs a very important technical function within any IT organization, managing and developing the corporate database so that it runs efficiently and effectively. DBAs should be technically highly skilled individuals who understand the inner workings of database systems. This takes extensive training and knowledge of the system upon which they are working. It is essential that the training of the DBA is regularly enhanced, since the technologies of the systems upon which they work continue to evolve. Similarly, the regulatory environment in which the corporation and its systems operate also changes and DBAs need to be educated in their regulatory requirements, e.g., Sarbanes–Oxley and HIPAA.

The role of the DBA is undergoing a change in terms of the nature of the databases upon which they operate. The advent of complex packages such as ERP systems requires specialized skill sets in order to be able to understand those systems and their database functionality. The databases of these systems are not intended to be managed in the same way as the traditional databases of the past. ERP databases are highly specialized and extremely complicated, so customization of their structures, even by experts, is highly ill-advised because the systems are self-configuring, requiring little outside intervention.

Summary of positive issues

DBAs act as the controlling authority for any corporate data, database, or system relating to data. DBAs are highly trained and skilled in database technologies. DBAs act to ensure the security of their database systems and to enforce regulatory requirements.

Summary of potentially negative issues

DBAs are required to be highly skilled and must have on-going educational training. Employment of a weak or under-skilled DBA can lead to catastrophic database problems. Some systems require very specific skills not typically found in a DBA, e.g., knowledge of ERP systems.

Reference

- C. Mullins (2002). *Database Administration: The Complete Guide to Practices and Procedures* (New York, Addison-Wesley).

Associated terminology: Data Protection Act, Sarbanes–Oxley, HIPAA.

Data-flow diagram

Foundation concept: Software development lifecycle.
Definition: Data-flow diagrams are graphical models that show the flow of data through a system, the external sources and destinations of data, the processes that transform the data, and the locations where the data is stored.

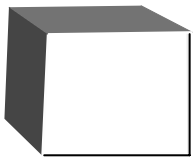
Overview

The concept of a logical data-flow model, usually known simply as a data-flow diagram (DFD), is used in the analysis and design phases of the system's lifecycle as a communication tool to understand, verify, and validate the system requirements.

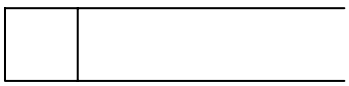
A DFD is constructed using the following symbols:

A process (e.g., check for stock availability, process credit-card payment):

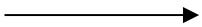
An external entity (e.g., customer, vendor):



A data store (e.g., customers, products, payments):



A data flow (e.g., credit-card information traveling between a customer and a payment process):



When performing process modeling (creating DFDs), the systems analyst starts at the highest level of abstraction, and once all the processes and data flows have been mapped at that level, the analyst would create a lower-level DFD for each high-level process by "exploding" the process to show greater detail (i.e., sub-processes and their data flows). This is repeated until all processes have been "exploded" and the lowest level of detail desired achieved, resulting in a set of "primitive" processes.

Traditionally DFDs were used to model both existing and proposed systems. Currently DFDs are used to model only the proposed system. Another difference between traditional and current modeling concerns the order in which data and process modeling are performed. Traditionally, using structured techniques, process modeling was performed prior to data modeling. However, today, due to the influence of information engineering, data modeling precedes process modeling.

Business value proposition

DFDs have been a popular and successful means used by systems analysts since the late 1970s. They provide an intuitive visual view of the system through "box-and-line"-based constructs as opposed to earlier text-based descriptions. These constructs can be easily interpreted by system users and facilitate the analyst–user discussion and communication necessary to understand and validate the system requirements.

The models can not only be used to create new systems but are also used in the redesign of business processes, and act as a basis for a business process re-engineering effort. DFDs are strongly associated both with the "*systems analysis and design*" approach to capturing system requirements and with the "*informal methods*" school of systems development. The approach is well known and widely used and has been a core element in the design of transaction processing systems since the 1970s.

Summary of positive issues

DFDs are a widely known means for modeling used by systems analysts in analysis and design. They are intuitive to use and are easily understood by non-technical personnel. This modeling method allows a high-level view of a system to be created and then exploded down to lower levels of refinement. The model fits well with informal design methods such as the *Waterfall method* and is a traditional central component of methodologies for systems analysis and design.

Summary of potentially negative issues

The data-flow method is becoming dated and is being superseded by models such

as *Semantic data models*, *Object models*, and the *Unified modeling language*, which facilitate the mapping of process models into constructs that are more rigorous and are easier to map onto more modern programming environments such as *Object-oriented programming*.

Reference

- T. DeMarco (1978). *Structured Analysis and System Specification* (New York, Yourdon Press).

Associated terminology: UML, Normalization.

Data mining

Foundation concept: Database.
Definition: Data mining is the technique of searching data for "patterns of interest" to the miner.

Overview

Data mining, sometimes referred to as *Knowledge discovery in databases* (KDD), is the use of algorithms to search through data sets looking for interesting patterns or confirmation of a predicted property. Data mining uses a complex set of techniques drawn from a variety of computer science disciplines, including *Databases* and *Artificial intelligence*, and from statistics.

The data upon which the mining activity is performed can be in any form; however, a *Data warehouse* in which transactional data has been cleaned, formalized, and archived can make the process more straightforward. In principle, data mining uses a variety of techniques from database theory and machine learning to examine data sets looking for patterns that display "interesting relationships." For example, a supermarket may establish the success or failure of a promotion campaign by examining the sales data, answering questions such as "did the target group buy more or less of the product?" Did they switch brands? Did they increase the volume of the product consumed, or did they buy more of another product in association with the promoted product?

The theory and practice of data mining continue to develop and evolve. A relatively new branch of data mining is *Text mining*. Much of the data on the internet and in databases is actually textual in nature rather than numeric (e.g., web pages containing normal English prose). This has led to a wide variety of techniques being used to analyze the text, its patterns, and its content. The goals for undertaking text mining are the same as those for quantitative data mining, namely the identification of new and novel data patterns within seemingly heterogeneous data.

Business value proposition

Data mining is the process of analyzing data by detecting patterns. For example, data mining performed by the WalMart company revealed that, when a hurricane warning is issued by the US Weather Service, demand for strawberry-flavored breakfast tarts is dramatically heightened in stores in the affected area ("Private FEMA," *Wall Street Journal*, September 10, 2005). Armed with this knowledge, WalMart can, immediately following (or even preceding) a hurricane warning, order, ship, and shelve more breakfast tarts, providing the customer with the product they desire in a just-in-time-delivery manner, driving greater profit for the company. Data mining in this instance directly contributes to WalMart's bottom line, allowing them to adjust their product-demand levels and achieve high completed-sales rates with a very short product shelf-life/cycle-time.

Data mining (including text mining) allows organizations to extract potentially rewarding information from their historical data sets. These techniques enable CIOs and organizations to leverage data sets that may have fixed storage costs associated with them due to compliance and regulatory issues.

Summary of positive issues

Data mining enables new and potentially important and profitable information to be extracted from a data set. The technologies enable a wide range of data sets to be examined, including text. Data mining can be used on traditional databases and data warehouses. Vendors are available to provide consulting, tools, and support. The area is supported by an academic and practical literature.

Summary of potentially negative issues

Data mining and text mining can be complex and challenging from the statistical and technological perspectives. Data mining can also be resource-intensive and requires a mixed-staffing initiative consisting of technologists and staff knowledgeable in the processes captured in the data sets. Data mining is still an emerging technology.

References

- K. Hirji (2001). "Exploring data mining implementation," *Communications of the A.C.M.*, Volume 44, No. 67.
- I. Witten and F. Eibe (2005). *Data Mining: Practical Machine Learning Tools and Techniques* (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Business intelligence, CIO.

Data pool

Foundation concept: Database.
Definition: A data pool is a data set specifically created to allow the data to be retrieved by third parties.

Overview

A *Data pool* differs from a standard database in that it is a data set created by an organizational entity, separate from its internally focused, live transactional systems, specifically for the purpose of allowing third parties data access. An example of this is the data pool operated by the National Snow and Ice Data Center (NSIDC), which archives and distributes data about snow cover, ice sheets, and related topics. The NSIDC has over 500 data sets that can be accessed online and retrieved via FTP, ranging from data on "Shallow borehole temperatures, Ilulissat, Greenland" to "Summer surface hydrographic measurements in the Laptev Sea, 1980 to 2000." A key aspect of data pools is that the data is reliable, clean, and has well-defined fields.

A second type of data pool can be defined as a large data superset comprised of data subsets containing the product meta-data of an individual company's products. The pressure for organizations to become more connected in terms of their supply chains has led many to adopt common technologies for communicating, such as the use of XML over the internet. However, for a large manufacturer, the task of notifying all of its customers of all its product changes or modifications each and every time a change is made would be a daunting task using a peer-to-peer messaging system.

The need to share data has led to the concept of a *Data synchronization hub*. Such a hub allows companies to place data pertaining to items in an accepted standard format such as *EAN.UCC*. The data is then placed in a data pool. This data pool is usually administered by a third party (but for large companies this can be done in-house) and information pertaining to the data set is sent to a *Master data registry* (MDR) such as the global synchronization network *GS1* or the *WorldWide Retail Exchange* (WWRE), which holds the information and records the location of each item's data pool.

Customers or other companies search an MDR through their chosen data pool provider and notify the data pool provider of the items they wish to subscribe to.

The MDR operator uses a *Synchronization engine* to automatically and continuously synchronize the data between the two companies. For example, a component manufacturer may place its product specifications in a data pool; then, when they change the specification of one component, all parties who have subscribed to the data pool will be notified that this change has occurred.

Business value proposition

Data pools allow organizations to simplify their data collection, modification, and dissemination functions. For a single entity providing third-party access to clean data this can be done straightforwardly through an internet-based FTP connection.

The use of data pools through data synchronization hubs allows organizations to manage their vendor–customer data relationships in a cost-effective, efficient, and timely way.

Summary of positive issues

FTP-based data pools allowing third parties to download data upon demand are relatively easy for individual entities to establish. For commercial organizations, standards such as "EAN.UCC" have been established, to which companies can map their data. Having achieved standardized data formats, there are various third-party data pool providers through which companies can manage their data-pool requirements. Several very large data synchronization companies have been established to manage the data requirements of large commercial organizations.

Summary of potentially negative issues

The conversion of data to the UCC standards can be a challenging resource-intensive problem.

References

- http://www.worldwideretailexchange.org/cs/en/index.htm.
- http://www.transora.com/.
- http://nsidc.org/.

Associated terminology: Client, Server, Data mining, XML.

Data Protection Act (UK only)

Foundation concept: Data.

Definition: The UK Data Protection Act defines the obligations for owners of personal information (the data processors) and the rights of the data subjects.

Overview

The Data Protection Act (DPA) has established a standard of good practice for "anyone processing personal information." It is based upon eight principles: the data must be

(1) fairly and lawfully processed;
(2) processed for limited purposes;
(3) adequate, relevant, and not excessive;
(4) accurate and up to date;
(5) not kept longer than necessary;
(6) processed in accordance with the individual's rights;
(7) secure; and
(8) not transferred to countries outside the European Economic Area unless the country has adequate protection for the individual.

These principles are combined with six conditions, one or more of which must have been met in order for the personal information to be considered fairly processed:

(1) the individual has consented to the processing;
(2) processing is necessary for the performance of a contract with the individual;
(3) processing is required under a legal obligation (other than the one imposed by the contract);

(4) processing is necessary to protect the vital interests of the individual;
(5) processing is necessary to carry out public functions, e.g., the administration of justice; and
(6) processing is necessary in order to pursue the legitimate interests of the data controller or third parties (unless it could unjustifiably prejudice the interests of the individual).

Sensitive personal data includes racial or ethnic origin, political opinions, religious or other beliefs, trade-union membership, physical or mental health concerns, sex life, and criminal proceedings or convictions. The data subjects have seven rights detailed under the DPA:

(1) the right to access their data;
(2) the right to prevent processing;
(3) the right to prevent processing for direct marketing;
(4) rights in relation to automated decision-making;
(5) the right to compensation;
(6) the right to rectification, blocking, erasure, and destruction; and
(7) the right to ask the Commissioner to assess whether the act has been contravened.

Criminal offenses are detailed for circumventing the act and include notification offenses, procuring and selling offenses, and electronic-communications offenses. The act is overseen by the Information Commissioner's Office, a branch of the UK government.

Business value proposition

The act enables businesses to operate under a regulatory framework and clearly sets out their obligations to the data subject.

References

- Data Protection Act 1998 Chapter 29.
- http://www.informationcommissioner.gov.uk.

Associated terminology: Law cross-reference.

Data quality audit

Foundation concepts: Database, Information lifecycle management.

Definition: A data quality audit is a check to ensure that a data set actually matches the data definitions originally established for it.

Overview

A data quality audit (DQA) is an assessment of a data set with respect to the original data definitions for that data set. Audits can be performed upon any data set, ranging from a spreadsheet to the database of an ERP.

Many organizations have been compiling data into databases for over 30 years, and many "legacy" databases continue to be used in corporate computing systems. The quality of the data entered into these systems is in many cases undetermined. For example, if a company has several databases scattered around its organizations, it may be that customers have different names or addresses recorded in different places: Bob Smith, R. Smith, and Robert Smith may all be the same person; Fort Lauderdale and Ft. Lauderdale may both be the same place. In order to prevent this type of situation, a database administrator may undertake a DQA.

One methodology for performing a DQA has been proposed by Gonzales, and it is representative of the methods in general. His model has six stages that are iteratively applied to different data categories (e.g., customer data, account data, and transactional data). Stage 1 is an examination of the data definition language associated with the data to be examined. Stage 2 requires the analyst to identify a sample data set, whose size will reflect the complexity of the database and its structures; this is necessary because some data sets

can be many terabytes in size and it would clearly be impracticable to survey the whole data set. Stage 3 involves the establishment of a base rule set to use in conjunction with the sample data set (e.g., the volume held in a tanker transporting fuel oil can not be less than zero or greater than the tanker's capacity). Stage 4 involves expanding the rule set, establishing confidence levels, and designing the output report structures. Stage 5 involves encouraging the user community to examine and modify rules, recalculate the confidence intervals, and adjust any reports that will result from the audit. Finally, in stage 6 the analysis is performed, and results are collected and examined.

A poor result in a DQA should lead an organization to re-examine the data-entry and process models around which the systems are based. It is possible to clean up the data and re-establish data integrity, but this can be a difficult and expensive proposition that may also involve cutting out data that is out of bounds and can not be corrected through knowledge available from other data sources.

An alternative to modifying a data set is to establish a new data set through a process known as *ETL* (extraction, transformation, and loading of data), a process that is usually related to the establishment of a *Data warehouse*.

Business value proposition

DQAs are essential components of a corporation's data management process. The older the data set, the higher the probability that it contains many errors. Modern information systems such as ERP systems, which have only one instance of a data item in its singular database, attempt to minimize the opportunity for data corruption to occur, both in the form of data-entry constraints and through detecting errors in the applications themselves.

The creation of a new data warehouse or the establishment of an ERP system allows the historical data to be dumped and written off or to be scrubbed through the ETL process; this is a great opportunity to re-establish the integrity of the data set.

Summary of positive issues

DQAs facilitate the establishment of statistical data showing the quality of a data set. Tools, methodologies, and consultants are available to support the DQA process.

Summary of potentially negative issues

A large amount of resources may be needed to audit large old data sets and systems. Syntactic errors in data are easier to identify than semantic ones. Data sets are no longer just numerically based and may include heterogeneous data sets, e.g., XML files, pdf files, graphical images, and audio files, which are more difficult to audit.

Reference

- M. Gonzales (2004). "The data quality audit," *Intelligent Enterprise*, July.

Data warehouse

Foundation concepts: Database, ERP, File server.

Definition: A data warehouse is a file server that contains data that has been extracted, transformed, and loaded into it from other databases. The data is then accessible for analytic programs, commonly known as business intelligence software, to run upon it. The results are frequently presented in a graphical format.

Overview

The term *Data warehouse* is used to describe a dedicated facility for aggregation and storage of data. Data warehouses are repositories of data that has been extracted from corporate systems. The data undergoes a transformation process (or is *Scrubbed* as it is sometimes called) in which the data is checked for type and consistency, before it is loaded into the *relational database* of the warehouse where the information

is stored. This process is known as *ETL* (extraction, transformation, and loading). Once the data has been loaded, it is available to be manipulated and examined so that analysis may be performed upon it.

Data warehouse analytic software allows the data to be examined from multiple perspectives, termed *Multi-dimensional data views* (or *Slicing and dicing*). This analysis is also frequently termed *Cubical analysis* since the data is often manipulated into a three-dimensional cube, but more dimensions are possible. For example, a data warehouse for a computer manufacturer may be structured in three dimensions, of which the first is a dimension that is based upon the sales markets: north, west, south, and east. A second dimension may be based upon product type (PC, PDA, laptop, and server), while the third dimension may be sales by quarter. This would provide the basis for data analysis by "slicing and dicing" these three dimensions. The programs that run on the data warehouse allow the analyst to drill down on the data, perhaps starting with an analysis of sales for all the markets the company serves over one year, then drilling down to a quarter sales period, then drilling down to a given month, and then again down to a given week or day.

Business value proposition

The use of a data warehouse allows a business to have a central repository for historical data. A data warehouse removes the need to keep the data in other online systems such as the organization's ERP. The data warehouse facilitates the backup and security of the data. The data after ETL will be consistent, verified, and archived. The executive information system software or *Business intelligence* software that runs on the data warehouse allows the data to be examined at multiple levels of detail and the results to be presented in a variety of formats, including graphical ones. The ability to quickly manipulate data in this way allows for focused one-time inquiries, with trends and exceptions being processed easily, and thus facilitates rapid decision making. This type of analysis would be expensive and very difficult in traditional databases.

Summary of positive issues

Data warehouses are secure data repositories whose data has been "scrubbed" and cleaned, resulting in a database with high integrity. The data structure used to store the data, namely the data cube, facilitates rapid data analysis and a highly flexible approach to business intelligence querying.

Summary of potentially negative issues

Data warehouses require dedicated servers if they are to be most effective and the data extraction, transformation, and loading (into the data cube itself) can be difficult, thus a careful return on investment analysis needs to be undertaken when developing such systems.

References

- W. Inmom (1996). *Building the Data Warehouse* (New York, John Wiley and Sons).
- E. Vitt, M. Luckevic, and S. Misner (2002). *Business Intelligence* (Redmond, WA, Microsoft Press).

Associated terminology: Business intelligence, Data mining, Database, OLAP, ETL.

Decision support system (DSS)

Foundation concept: Business intelligence.
Definition: A decision support system is an application through which managers and executives can analyze corporate data.

Overview

Decision support systems (DSSs) are the predecessors of what has become known

as *Business intelligence* or *Business analytics*. They consist of three components, a dedicated data warehouse, a modeling system, and an interactive user interface, usually with graphical capabilities. DSSs are also the basis of *Executive information systems*, analytic systems developed to address the needs of senior executives, with the potential to display results in the form of an "executive dashboard."

A collaborative form of DSS, known as *Group decision support systems* (GDSSs) has also been developed. The GDSS concept evolved in the 1980s and has also been known as *War rooms*, *Decision support laboratories*, and *DDS conference rooms*. GDSSs help managers or executives to solve a problem by bringing them together, either physically or virtually, to make decisions that are supported by their colleagues and by the most appropriate data and analytic tools available.

Business value proposition

DSSs have evolved from the academic-based arena into the commercial world, and have become known as business intelligence or business analytics; they are used in conjunction with ERP and data warehouse systems.

The field of GDSSs also continues to evolve, but is mainly situated in academic forums. The use of a GDSS allows a large group of people focused upon a particular problem to be connected together. Led by a human facilitator, the members of the group interact with the system simultaneously, developing ideas that the system can filter, sort, and analyze. The group, in conjunction with the facilitator and the system, then acts to rank and develop the ideas further. The systems may have the ability to make the input from each participant anonymous, which can strengthen the arguments, overcoming the problem of rank or dominant group members overwhelming a discussion.

Summary of positive issues

DSSs allow individuals or groups to consider a problem through the use of a computer system that provides the decision makers with data, analysis tools, and reporting capabilities. This allows better decision-making within information-rich environments or when critical analysis of multiple data sets is required in light of given decision criteria.

Summary of potentially negative issues

While military and government institutions have adopted the technology, the expense and dedicated resources sometimes required have limited their adoption within the wider commercial world, so DSSs tend to reside in academic and research institutions.

Reference

- D. Power (2003). "A brief history of decision support systems," http://DSSResources.COM/history/dsshistory.html.

Associated terminology: Business intelligence, Data warehouse, ERP.

Denial-of-service attack

Foundation concepts: Security, Network.

Definition: An attack that is intended to prevent legitimate users from accessing or having full use of a computer system, rather than attempting to destroy, steal, or modify information.

Overview

Viruses, worms, and Trojan horses have a clear malicious intent; they attempt to destroy or disrupt computer resources. Spyware attempts to steal information, and adware hijacks a computer and uses it as an advertising outlet. A denial-of-service (DoS) attack is a fourth form of digital assault. It does not attempt to cause permanent damage or to steal anything, just

to prevent proper use of a system. Because of this, many self-righteous DoS attackers manage to convince themselves that they are not really doing anything illegal.

A DoS attack is often designed to exploit a known error in a popular operating system, and, since there are so many known errors in popular operating systems, there is a very wide variety of attack methods available to the attacker. A very popular attack a few years ago was known as the "ping of death," in which a deliberately invalid message was sent using the ICMP protocol, and, due to a bug in the operating system, receiving that message would immediately crash the target computer. No permanent harm done, just reboot and it's running again, but all it took was a very simple program, easily downloaded from the internet, to send the same message automatically every 5 minutes, rendering the target computer completely useless. To this day, many installations have disabled their "ping" service even though the bug has long since been fixed. Many responses to attacks seem to be based more on superstition and voodoo than on sense. Some attacks make use of known bugs in networked applications, where reception of a particular message might result in the target application filling a whole disk with bogus files, or embarking upon a nonterminating computation.

Another form of DoS attack requires no knowledge of any operating-system flaws at all. Simply flood the target computer with as many internet-connection requests as you can send; it will be unable to process them all, so normal requests from legitimate users will be crowded out.

A common means of mounting DoS attacks untraceably is to take over an unwitting third-party computer, or preferably a lot of them, through virus and Trojan-horse software, and use those computers to act as an army of *Zombie computers* sending the flood of messages. This is called a *Distributed denial-of-service attack*, and is especially effective because hundreds of computers can be used in unison to create an unsurvivable flood on a whole target network. The messages that cause a flood attack can be traced back to their source, but the source often turns out to be an innocent third party's unsecured home computer, which may have seemed to be running quite slowly, but would have shown no other sign of having been compromised.

Business value proposition

There is no positive aspect for businesses regarding a DoS attack; the only thing that organizations and individuals can do is guard against them. This entails having a strong security policy in place and understanding what the attacks are and how they break into an organization or take over a computer.

A strong and continuously updated firewall system is the best form of defense, which should be combined with regular systems checkups to ensure that the system has not been taken over and become a zombie controlled from outside the organization. While it is important that organizations maintain an understanding of the latest DoS strategies employed and defend themselves against them, it is also important that, when a technique has been defeated, the company assess whether systems "lockdowns" are still necessary. Such was the case when the "ping of death" was a peril to systems; organizations permanently closed down their "ping" service, denying access to a useful utility, even though the bug through which the DoS attack was being made has subsequently been resolved.

Summary of potentially negative issues

DoS attacks can be problematic for organizations and result in systems outages. These outages may be prolonged and persist until a fix has been found by the operating systems vendor, security software

vendor, or other security specialist, depending upon the type of attack.

Reference

- J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher (2004). *Internet Denial of Service: Attack and Defense Mechanisms* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Virus, Trojan horse.

DHCP (Dynamic Host Control Protocol)

Foundation concepts: Internet protocol, Network.
Definition: A system for temporarily allocating IP addresses to computers on an as-needed basis.

Overview

Every computer or network-enabled device connected to the internet must have an *IP address* (internet-protocol address). An IP address is four small numbers separated by dots (such as 221.39.107.82), which uniquely identify the computer, distinguishing it from all others. All information transmitted over the internet uses IP addresses to specify its destination; a computer without an IP address will be unable to receive or transmit anything.

Long runs of consecutive IP addresses are allocated by a central authority (see *Domain name*) to individual organizations, and those organizations are responsible for allocating the individual IP addresses to the individual computers within their domain. In the early days of the internet, this scheme worked very smoothly, but demand has reached such levels that there are not enough IP addresses to go round. *IPv6* (the new version of the internet protocol, in which addresses are much longer) will eventually relieve some of the problems, but is still not in general use.

DHCP, the Dynamic Host Control Protocol, is a very popular solution. An organization may have enough IP addresses allocated to it to cover the number of computers and other network devices that are normally connected at any one time, but not enough for every single device that they are responsible for. This is a very common situation for internet service providers (ISPs), for which only a fraction of their customers will be actually using the internet at any given time.

When a DHCP-enabled device first attempts to access the network, it must negotiate with a DHCP server for an address. If there is an IP address available, the DHCP server will allocate it temporarily to the requesting device, giving it a *Lease* on the number with a specified expiration time. The device may then use that IP address as its own. When the device disconnects, or when the lease expires, the DHCP server simply reclaims the allocated IP address, and the device's network communications stop working. A device may request a lease renewal at any time before expiration, and it is granted if demand is not too high.

DHCP allows a large set of computers to share a not-so-large set of IP addresses, but is successful only if demand for addresses does not exceed the number available. For end-user systems DHCP provides a generally satisfactory service, but it is not at all suitable for systems that play host to permanent or long-term network services, such as web servers and email servers. In order to be found by their clients, servers must have a known, and therefore non-variable, IP address.

Network address translation (NAT) is another technology that may be used either in conjunction with, or instead of, DHCP; it helps to solve the problem of having too few IP addresses available to meet demand.

Business value proposition

Network administrators use DHCP to overcome the problem of not having sufficient IP addresses available to them to allocate one IP address per device. Thus, assuming that the network is to be connected

to the outside world, the network administrator will take the addresses available and allocate one or more of these addresses to a specific server that is connected to the internet or external network. This server then acts as the gateway for internet traffic. The remaining IP addresses can then be allocated to other servers, known as DHCP servers. These servers operate over subnets, whose clients may request an IP address. These are then given to the client on a temporary basis, known as a lease, and, when the lease is about to expire, the client can then request an extension to that lease.

The DHCP servers allocate, maintain, and service requests across the network in a way that is efficient, cost-effective and resource-preserving. Large IP address blocks are extremely difficult to obtain and are costly when available, especially in those countries which were early IP adopters. DHCP also frees the network administrator from having fixed network addresses for each client in their network, thus facilitating the management of the system, and enabling devices and systems to be added without worrying about re-allocating IP addresses.

Summary of positive issues

DHCP is a cost-effective method of allocating IP addresses to devices on a corporate network. DHCP is well known and easy to implement, and the operating systems that run servers usually have DHCP capabilities within their toolsets.

Summary of potentially negative issues

Networks that have heavy traffic from a large number of devices that are continually online may place demands upon the DHCP servers that result in some devices being without an IP address and slowed networks due to the volume of device-generated traffic. DHCP is not suitable for devices that require specific static IP addresses.

Reference

- R. Droms and T. Lemon (2002). *The DHCP Handbook* (Indianapolis, IN, Sams).

Associated terminology: ISP, Protocol.

Digital

Foundation concepts: Bit, Binary.
Definition: A digital system is one that is based upon a sequence of events, symbols, or data.

Overview

Digital data simply means data represented as a sequence of symbols, rather than as a direct physical analog. For example, the length of a pencil could be represented as a proportional voltage, with one volt for one inch; that would be an analog representation. Alternatively, one could measure the pencil and write down its length, 6.31 inches; that is a digital representation.

Working directly with analog representations may seem more natural and accurate, and indeed this technology was invented long before digital computers, but analog computation has severely limited accuracy, speed, and reliability. Today, digital computation is universally superior.

Anything written as a series of symbols (e.g., 6.31, "cat," 今天的天氣相当不错) is already in digital form, but an additional very simple encoding step (e.g., c → 01100011, a → 01100001, t → 01110100) is usually taken to reduce the symbols to binary numbers, which are then concatenated to produce a pure bit-stream (e.g., "cat" → 011000110110000101110100), which is the internal form used by a modern digital computer. Every measurable quantity can thus be reduced to a stream of 0s and 1s.

Even something as seemingly non-digital as music is simple to digitize. Sound is nothing but a varying pressure wave in the atmosphere; the changing pressure may be monitored and digitally measured at

frequent intervals, producing a stream of numbers that represents the continuously varying sound pressure. If the samples are taken frequently enough (44 000 per second is usually considered sufficient), they can be played back through a digital-to-analog converter connected to a loudspeaker, and the human ear will be unable to detect the difference.

Business value proposition

In practicality, all modern business computing systems are digital and based upon the binary representation. This allows the builders of electronic devices to ensure the successful interconnection and interoperability of devices.

Summary of positive issues

Digital computer systems based upon base-two computation are the standard mechanism for representing data.

Summary of potentially negative issues

Digital base-two representations, bits, have required a set of mechanisms to represent and manipulate the data at a higher level such that the data can be understood by humans.

Associated terminology: ASCII, Audio, Video, Compression.

Digital cash

Foundation concepts: Encryption, e-Commerce, Security.

Definition: A means of providing anonymous payment electronically.

Overview

Even in this age of credit cards and electronic commerce, cash still plays a very important role in society. Cash provides privacy, allowing people to buy and sell things without anyone else finding out. All other forms of payment (excluding barter, and cash is really just a token of barterable value) are traceable, and leave a trail of records that allow minute details of a private life to be reconstructed.

In many ways, the abolition of cash might seem to be a good thing: the practitioners of kidnapping, bribery, and blackmail would be dealt a serious blow if there were no untraceable forms of currency, and since 2001 the public has shown a remarkable level of acceptance toward the loss of traditional privacy rights. However, it should be borne in mind that cash-like things can not be abolished; if there were no such thing as cash, kidnappers and crooked politicians would simply demand payment in gold or some other substance with intrinsic value.

Digital cash would certainly be a popular invention: something that exists only in digital form, so it can be carried around or transmitted instantaneously between computers in any quantity without effort. It could be kept encrypted until it is needed, so the possibility of theft would be much reduced. It could be used to buy things without the payer's identity being revealed. It could be *backed up* like any other data, and so made immune to destruction by fire or other disasters.

Such benefits come at a price. Digital cash, in a form at least, has been successfully invented. It consists of a group of exceptionally complex (and patented) protocols based in part on public-key encryption and digital signatures. The fact that it has been known since 1990, and has not yet caught on, and in fact is still almost unheard of, is testament to the fact that it is not very convenient to use.

The complexity of digital-cash implementation is due to the seemingly impossible demands placed upon it. Once something is in digital form, it is impossible to prevent it from being duplicated. If somebody obtains a digital cash token, they could send that token to a hundred different vendors in payment for different

things. Since copying can not be prevented, security is instead based on preventing double-spending. Somehow, when a vendor receives a digital cash payment, they must be able to check immediately that it has not already been spent, and at the same time ensure that it can not be spent again in the future. To be truly cash-like, it would still have to be possible for the receiving vendor to spend it legitimately, so illegitimate double-spending is an extremely difficult problem to overcome.

Also, to prevent double-spending, all pieces of digital cash must in some way be different from all others. If they were not unique, it would be logically impossible to distinguish genuinely different tokens from illicitly double-spent copies. But, if all tokens are unique and digital, it would be possible for the issuer to record all tokens issued together with the identity of the person to whom they were issued, thus introducing traceability and missing the point of digital cash.

Business value proposition

The processes associated with implementing digital cash have been in existence since the 1990s; however, they have received very little publicity due to their complexity and the need to change the payment behavior of a large group of the population. The two types of digital cash, anonymous and identified cash, both have many issues associated with them.

Anonymous digital cash allows people to transfer monies without a trace being left, which would be a problematic issue for governments that rely upon a paper trail to prevent unlawful activities. Physical cash makes it hard for a criminal to carry around millions of dollars, pounds, or yen without attracting attention.

Identified cash has problems associated with it that stem primarily from the fact that every transaction would be traceable, and complete profiles of an individual's spending habits could be created. While governments may enjoy the ability to identify all taxable incomes, the impact that such a transaction trail would have upon society has limited the political will to implement such systems.

The current limits of deployment of digital cash include the implementation of systems for electronic payment from bank to bank (e.g., to pay a water bill or the mortgage), which are in effect wire transfers, systems for electronic payment from bank to a credit card (e.g., this can be used to make payments related to online auctions), digital-cash-based debit cards (e.g., in the United States welfare recipients use electronic benefits transfer (EBT) systems in which a debit card has replaced the use of paper "food stamps"). However, none of these systems approach true digital cash and the probability of large-scale use of such systems in the near future is low.

Summary of positive issues

Digital cash provides the ability to transfer money electronically from one person to another. The use of identifiable digital cash allows governments and individuals to trace payment trails. Anonymous digital cash can reduce the complexity of transactions, and increase customer confidence.

Summary of potentially negative issues

With identified digital cash governments can monitor all digital transactions, and small payments may incur disproportionate transaction costs. With anonymous cash the lack of a transaction trail may facilitate illegal activities. There are significant problems in developing processes to prevent double-spending, ensuring proof of ownership, and backing up cash, and as a result the digital cash protocols are extremely complex.

References

- D. Chaum (1983). "Blind signatures for untraceable payments," in *Advances in*

Cryptology CRYPTO '82, Lecture Notes in Computer Science, ed. D. Chaum, R. Rivest, and A. Sherman (New York, Plenum Press).
- D. Chaum (1985). "Security without identification," *Communications of the A.C.M.*, October.

Associated terminology: Cracking, Protocol.

Digital certificate

Foundation Concepts: Digital signature, Public key, Encryption.
Definition: A public key, digitally signed by a trusted authority.

Overview

The only real problem with public-key encryption, once the strength of the algorithm has been established, is key management. As well as the obvious need to keep private keys utterly private, there is a correspondingly urgent need to make public keys utterly public.

If a criminal can trick anyone into believing that a public key generated by him is in fact your public key, then, in the digital world, he can *be* you. Encrypted messages intended for you will be readable by him and only him, and, possibly worse, he will be able to sign legal documents (contracts, bank drafts, confessions) *as* you.

This is why wide publication of public keys is essential. Everybody you do business with should ideally have a secure copy of your public key as provided directly by you; everyone you might do business with in the future should have instant access to an incorruptible public-key directory service. Of course, this ideal can not practically be realized. If you had to deliver your public key to everybody in person, electronic commerce would be pointless, and how would they verify your identity even then? With a public directory, it would always be possible for someone to subvert a communication channel, temporarily redirecting accesses to the public directory to a fraudulent one of their own.

Digital certificates are an attempt to solve this most serious problem. The underlying idea is that, if there is one entity anywhere that can be fully trusted (perhaps a saintly person with a lot of security and computing power, perhaps a trustworthy corporation!, or even a trustworthy government!), they would become a *Certification authority* (CA). Everybody, on first receiving their own public-key–private-key pair, would register their public key with the CA. The CA would very thoroughly investigate the individual to verify their identity, doing at least as much as modern governments do before issuing passports. Then the CA would then issue the applicant with a digital certificate, a simple and short document saying essentially "This . . . is the true public key for . . ." The certificate is encrypted using the CA's own private key.

There are two essential ingredients to this scheme. The first is that everybody in the world should know the CA's public key; it should be stored indelibly and immutably on all systems, and never need to be looked up from any directory service. The second is that the CA must be perfectly trustworthy and perfectly secure. A dishonest CA could get away with almost anything. If the CA's private key is ever revealed then all identities are open to theft.

If the CA is perfectly trustworthy and perfectly secure, then digital certificates are as reliable as the encryption system used. Nobody can ever present a fraudulent public key to steal another's identity, because nobody would ever accept a public key that does not come as part of a digital certificate. Nobody can make a false digital certificate without access to the CA's private key.

Secondary CAs may also be created, and would most probably be necessary since the

load on a single universal CA would be overwhelming. Each secondary CA would have its own digital certificate of identity signed by the one primary CA. Then every digital certificate issued by the secondary authority would contain an extra clause that is a copy of the secondary authority's own digital certificate. That means that those receiving a certificate issued by a secondary authority may still validate it with knowledge only of the primary CA's public key.

Currently, there are some organizations offering competing services as trusted CAs, and their public keys are not securely built into computer systems. Digital certificates are offered for costs in the range of hundreds of dollars. It is envisaged that each organization would obtain a single digital certificate from a central authority, and then act as its own secondary certification authority for internal purposes.

Business value proposition

Digital certificates are a means that allows companies or individuals to authenticate the entities (e.g., companies, people, and governments) that they interact with online. There are two primary options for businesses wishing to deploy digital certificates; they can administer their own certificates, or they can outsource the management of the certificates.

It would be difficult for most organizations to administer and issue their own digital certificate. Also they would have the problem that their certificate would be doubted by many, if not all, of those who received it, hence the advent of CAs. The existence of a CA allows organizations and individuals to make use of an established certification process. Further, these authorities usually provide a suite of applications that work in conjunction with software from other vendors (e.g., web-server vendors) to provide a secure cost-effective mechanism for certificate management. The outsourcing of the certification process also results in more efficient certification management and a potential for reducing the total cost of ownership involved in the certification-related processes.

Digital certificates and associated applications can be used for a variety of purposes, including the certification of identities, adding encryption, confirming sources of data/information, and restricting access to information on the basis of certified identify.

Summary of positive issues

Certificates are easy to create and use through CAs. They provide positive identification of data authors, individuals, companies, or other entities.

Summary of potentially negative issues

Great care must be taken in selecting a CA: anybody can set up a web site, call themselves a CA, advertise on the internet, and sell digital certificates. A CA that is not *completely* trustworthy is much worse than no CA at all. A prospective customer must ask themselves what makes this private company so thoroughly worthy of trust, and how can they guarantee complete security of their encryption keys?

Digital certificates are currently a very expensive proposition, especially for individuals and small organizations. It may be hoped that the advent of secondary CAs could ease the situation, as could government regulation in this commercially vulnerable area which impinges directly upon national-security concerns.

References

- B. Schneier (1996). *Applied Cryptography* (New York, John Wiley and Sons).
- J. Feghhi and P. Williams (1998). *Digital Certificates: Applied Internet Security* (New York, Addison-Wesley).

Associated terminology: Phishing, Cracking, Hacker.

Digital Millennium Copyright

Definition: The Digital Millennium Copyright Act of 1998 is divided into five titles that implement two 1996 World Intellectual Property Organization (WIPO) treaties, covering the copyright of performances and phonograms, liabilities for online copyright infringement, computer maintenance competition assurance, and issues such as web-casts and the design of ships' hulls.

Overview

The five titles of the act cover a wide range of technology copyright areas. Title I is the "Copyright of Performances and Phonograms Treaties Implementation Act of 1998." Title II is the "Online Copyright Infringement Liability Limitation Act." Title III is the "Computer Maintenance Competition Assurance Act. Title IV contains six "Miscellaneous provisions," while Title V is the "Vessel Hull Design Protection Act."

Title I implements two WIPO treaties: the WIPO Copyright Treaty (WCT) and the WIPO Performances and Phonograms Treaty (WPPT). The intent is to obligate WIPO member states to protect copyright works from being infringed upon by citizens of other member countries. Title I also obligates member states "to prevent circumvention of technological measures used to protect copyrighted works, and to prevent tampering with the integrity of copyright management information."

Title II acts to create limitations of liability for online service providers pertaining to copyright infringement; these include provision of data and materials in terms of "Transitory Digital Network Communications," "System Caching," "Information Residing on Systems or Networks at the Direction of Users," and "Information Location Tools."

Title III covers exemptions for the authorized copying or adaptation of software during the maintenance or repair of a computer by software owners or lessees.

Title IV covers special issues such as "Ephemeral Recordings for Broadcasters," "Distance Education Study," "Exemption for Nonprofit Libraries and Archives," "Webcasting Amendments to the Digital Performance Right in Sound Recordings," and the "Assumption of Contractual Obligations upon Transfers of Rights in Motion Pictures."

Title V pertains to the protection of hull and deck designs of vessels no longer than 200 feet.

Business value proposition

The act requires businesses to operate under a regulatory framework and within the context of a state that also works to protect the interests of copyright owners.

Reference

- "The Digital Millennium Copyright Act of 1998," The United States Copyright Office, 1998.

Associated terminology: Caching, Network, Law cross-reference.

Digital signature

Foundation concepts: Security, Encryption, One-way hash, Public key.

Definition: An annotation added to a digital document by an individual, that can not be copied onto another document or created by any other entity, and does not permit the document to be altered afterwards. An idealized version of the traditional pen-and-ink personal signature.

Overview

The idea of a signature is well understood. It is a personal mark that attests to the authenticity of a physical document, and which often constitutes some legally enforceable guarantee. The problems are manifold: documents may be altered after they have been signed; a signature may be copied onto another document; the signer

may falsely claim that a true signature was forged.

The notion of a digital signature as a digitized version of a person's normal signature, which may be included as an image in a document, is patently absurd. Digital documents are even easier to modify than paper ones, and a signature graphic can be cut out of one document and added to another seamlessly in seconds. "Digital signature" should not be confused with "electronic signature," which is simply the system to capture digitally the pen strokes made when signing for a paperless credit-card purchase with a stylus and electronic pad.

The true concept of a digital signature relies upon two essential technologies: *Public-key encryption* (q.v.) and *One-way-hash* operations (q.v.). To summarize, in a public-key encryption system, every individual has two encryption keys, one of which is completely private and secret, whereas the other is totally open and public. Data encrypted with one of the keys can only be decrypted with the other. A one-way hash is an operation applied to any data that reduces it to a moderately sized number (a few dozen digits); this number acts as a fingerprint for the data, so that the slightest change to the data would produce a totally different fingerprint number, but the fingerprint does not contain enough information to reconstruct any part of the original data.

A digital signature for a document is created by applying a one-way hash to that document, encrypting the resultant fingerprint using your private key, and appending the result to the document. It is a simple, fast, and secure procedure.

Anyone receiving this signed document can easily calculate what the fingerprint should be: they have the document, and one-way hash procedures are always openly published. They can also easily decrypt the copy of the fingerprint appended by the sender: public keys are always openly published. If the two versions match, the document is genuine.

The sender can not later deny having signed the document. The fact that the fingerprint was successfully decrypted using their public key is proof that it was originally encrypted using their private key, and nobody else has access to that.

The document can not be modified after it has been signed, because any change to the document will result in a change to the fingerprint (one-way hash), and the fraudster can not substitute a new fingerprint, because it must be encrypted with the sender's private key.

Digital signatures are useful for resolving intellectual property and non-disclosure disputes, since they enable individuals to prove that they know something without revealing what it is until later. Essential information may be recorded in a document and digitally signed in the normal way. The digital signature alone, without the document that it applies to, is sent to the other party. They can not reconstruct the document from the signature alone, but the sender is also incapable of constructing another document later that would match that signature. This mechanism also allows a third party to witness a document with their own digital signature without being able to read any sensitive information it may contain.

The whole system of digital signatures relies upon complete openness. All of the procedures must be public, otherwise nobody will be able to verify signatures; every participant's public key must be published as widely as possible, otherwise it becomes possible for fraudsters to extend fake public keys for their victims. The only thing that must be kept secret is the private key. There must be no possibility of a private key ever being revealed under any circumstances. The advance of electronic commerce almost equates private key with personal identity.

The DSA (Digital Signature Algorithm) is an embodiment of the DSS (Digital Signature Standard) produced by the NIST (National Institute for Standards and Technology) under authority of the Computer Security Act of 1987. It is based on a variable key length of between 512 and 1024 bits, which would be very large for normal encryption, but has attracted criticism for not being long enough for such a critical purpose. Some cryptanalysts are suspicious of the DSA, because it was designed by the NSA (National Security Agency), and not everyone is totally willing to trust them.

Business value proposition

The United States enacted a variety of laws pertaining to digital signatures, including the Electronic Signatures in Global and National Commerce Act of 2000 which aims to "*facilitate the use of electronic records and signatures in interstate and foreign commerce by ensuring the validity and legal effect of contracts entered into electronically*" (www.ftc.gov). At the state level, the Uniform Electronic Transactions Act of 1999 is intended to "*remove barriers to electronic commerce by validating and effectuating electronic records and signatures*," and the Digital Signature and Electronic Authentication Act of 1998 (SEAL), which is an amendment to the Bank Protection Act of 1968, is intended primarily to enable the use of electronic authentication techniques by financial institutions.

In 1998 the European Union (EU) issued an "Electronic Signature Directive" (1999/93/EC), which defines two different electronic signatures, namely the Basic Electronic Signature which they define as "*Data in electronic form which are attached to or logically associated with other electronic data and which serve as a method of authentication*" and an Advanced Electronic Signature, which must meet the following requirements: "(a) it is uniquely linked to the signatory; (b) it is capable of identifying the signatory; (c) it is created using means that the signatory can maintain under his sole control; and (d) it is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable." The Basic Electronic Signature is a general definition that covers digitally recorded hand-written signatures; it is the Advanced Electronic Signatures that correspond to the true digital signatures discussed above. Many EU countries have subsequently created their own legal frameworks in association with the EU directive, including the UK implementation of the directive termed the "Electronic Signatures Regulations 2002."

The US and EU laws principally aim to ensure two things: that the signatory is who they say they are, and that the digitally signed document is authentic. These two aspects of signed digital documents form the basis of a "non-repudiation" service, a service that can provide proof of data origin, transmission, and delivery, protecting the sender and receiver from false claims.

Summary of positive issues

Digital signature technology is well established and backed by legal frameworks in many countries. Non-repudiation services exist to validate data transfer.

Summary of potentially negative issues

The terms "digital signature" and "electronic signature" are sometimes used interchangeably when they may in reality represent different levels of technology and security. Not all countries have legislation to cover the use of digital signatures and country-specific contract law needs to be examined prior to commencing electronic transactions.

References

- B. Schneier (1996). *Applied Cryptography* (New York, John Wiley and Sons).
- Directive 1999/93/EC of the European Parliament and of the Council of 13th December 1999 on a Community

framework for electronic signatures, *Official Journal of the European Communities*, L13/12, January 2000.

Associated terminology: Digital cash, Digital certificate.

Digital wallet

Foundation concepts: Encryption, Security.
Definition: A secure software mechanism for storing credit-card and other payment information on a computer to ease online purchases and transactions.

Overview

A *Digital wallet* is an application that resides on a computer and manages personal information such as name, address, credit-card details, and bank-account information. This application is used to populate automatically the fields of an electronic web-based form such as those used when an online purchase is made. This relieves the user of continually entering the same data into forms. The data is encrypted and requires the vendor to send a digital certificate to verify their identity. Some digital wallet services are provided by third-party vendors and the wallet resides on their server.

Business value proposition

Digital wallets provide users with a secure and convenient mechanism by which to populate the fields of any web service they wish to interact with.

Summary of positive issues

Digital wallet software is available from several vendors, some of which hold the data on their servers for added security and ease of access; in that case users don't have to be at their own personal machine to use the resource.

Summary of potentially negative issues

The data in a wallet must be very carefully secured. Users may feel uncomfortable about having their personal financial data held by a third party.

Associated terminology: Digital certificate, Digital cash.

Disclosure and interception of

Definition: The Unlawful Access to Stored Communications Act (18 USC 2701), Voluntary Disclosure of Customer Communications or Records Act (18 USC 2702), and Wiretap Act (18 USC 2511) provide a general prohibition against intercepting communications or disclosing communications that one has valid access to.

Unlawful Access to Stored Communications Act, 18 USC 2701

This act makes it an offense for an individual who "intentionally accesses without authorization a facility through which an electronic communication service is provided; or intentionally exceeds an authorization to access that facility and thereby obtains, alters, or prevents authorized access to a wire or electronic communication while it is in electronic storage in such system."

Voluntary Disclosure of Customer Communications or Records Act, 18 USC 2702

This act covers electronic communications and the contents of those communications, e.g., voice and data. Its general purpose is to forbid those who provide public communications services from releasing communications to any but the intended recipient, thus providing some guarantee of security, privacy, and confidentiality.

It has three major sections. The first forbids providers of public electronic-communications services to knowingly divulge to any person or organization the contents of a communication while it is in electronic storage.

The second section extends coverage to those who provide any remote computing

service to the public (such as web-hosting services and data processing bureaus), similarly forbidding the release of communications either carried or stored by that service. The third adds a prohibition against the providers of communications and remote computing services releasing any information about subscribers or customers to any government entity.

While these aspects of the act cover the privacy framework for electronic communications and the contents of those communications, there is an extensive set of exemptions built into the act, some of which are necessary for business to occur (for instance, an email provider must be able to send email out of a mail box to its intended recipient). Other exceptions allow federal, state or local-government access in certain circumstances, such as the need for information in the case of life-threatening emergencies.

Wiretap Act, 18 USC 2511

The act is primarily intended to cover the intentional interception, use, or disclosure of any communication transmitted over a wire, orally, or electronically. The law also makes it an offense to use any device to intercept communications. This aspect of the act can be used to cover traditional "bugs," "wire taps," and eavesdropping, and to prosecute the malicious use of unauthorized "packet sniffers" (programs that can examine network traffic, looking for passwords or other information). The act also makes it illegal to disclose any information so found, for example placing a corporate password obtained through a packet sniffer on a "cracker" website. Subsequent to that it is also illegal for information to be taken from that cracker web site and used, knowing that the information itself was obtained illegally.

The act does describe a large number of situations in which the interception, use, and disclosure of information from wire, oral, or electronic sources are permitted, such as the activities of an officer or agent of the US government in the normal course of his official duty to conduct electronic intelligence, e.g., the FBI's Carnivore system used from 1998 to January 2005, which sifted through email at ISPs, capturing emails for only those individuals named in a court order.

The Cyber Security Enhancement Act, Section 225 of the Homeland Security Act of 2002, which amends various sections of 18 USC

This act takes the form of a set of amendments to previous legislation related to security and privacy on electronic (computer) systems. The act strengthens existing laws in various ways, such as explicitly making it an offense to advertise online illegal devices, defined as "any electronic, mechanical, or other device knowing or having reason to know that the design of such device renders it primarily useful for the purpose of the surreptitious interception of wire, oral, or electronic communications" (18 USC 2512). Previously only "newspaper, magazine, handbill, or other publication" were specified.

Laws pertaining to ISPs and their ability to report issues concerning life-threatening behavior were amended to allow greater flexibility in to whom they can disclose the information (18 USC 2702). The laws pertaining to hacking were strengthened and now cover fatal injury as a result of hacking (18 USC 1030). Sentencing guidelines were substantially strengthened for some offences.

Associated terminology: Law cross-reference.

Disk

Foundation concept: Storage.
Definition: The primary device for long-term data storage.

Overview

A disk drive consists of some number, usually one to five, of exceptionally smooth metal disks with magnetizable surfaces, locked together and spinning at a high rate (often over 10 000 revolutions per minute) on a single central axle. Very fine *Read–write heads*, one per surface, move rapidly in and out and are able to reach any point on any surface very quickly. The heads are able to detect magnetic patterns already present on the disk surface, and to write new magnetic patterns on any region they pass over. In this way binary data is stored as microscopic magnetic domains all over the disk surfaces.

The rate of rotation of the disks is closely controlled, and never varies (changing the spin speed would be a very slow operation because of the rotational inertia to be overcome), but the read–write heads are mounted on delicate arms (looking something like tweezers) that can move to any radial position over the disk surface on demand. Data stored on a disk is organized into concentric circular tracks: to access data, first the heads are moved to the right radial position for the desired track, then it is necessary to wait until the right portion of that track rotates itself under the heads.

A typical disk in a PC has a usable radius of around one inch (or 3 cm), into which hundreds of thousands of tracks are packed. That requires exceptionally fine control in the positioning of the heads, and that is a significant factor both in the cost of disk drives and in the data access time. Incredibly, the heads can move from one track to another, and be ready to read or write, in just a few thousandths of a second. The speed of a disk is controlled by two components, the track-to-track *Seek time*, plus the wait for the desired data to rotate into position, known as the *Rotational latency*. This means that the access time for any given piece of data is very variable, depending on the disk's current position at the time access is needed. Access speeds are usually quoted as the average random access time, which is very easy to compute: for the average access, the time must be the seek time plus (on average) half the time for a single rotation. For a disk that rotates at 7200 rpm (a very common speed, and the speed is usually prominently advertised), the rotation time must be 8.33 milliseconds; so any claimed average access time that is not notably greater than half of that figure must be viewed with grave suspicion.

When a disk is "powered down," and not spinning, the read–write heads actually rest on the surface of the disk. As the disk speeds up, friction causes air molecules to be dragged around with the disk surface, and at full speed (usually around 65 mph) a few molecules actually get dragged under the read–write heads, and they ride on a cushion of air just a few molecules thick. That is how they maintain the right distance above the surface and move without causing wear. It is also the main weakness of disks. If a disk drive receives some physical shock while it is in operation, the heads can crash into the surface and cause significant damage. This is the condition known as a *Head crash*. Usually the disk drive as a whole will survive, but the data stored in the damaged region can be irretrievably lost. Modern disk drives contain an automatic mechanism that ensures that the heads are moved to a designated *Landing zone*, a region of the disk not used for data storage, whenever power is lost, so they will naturally and gently land in an unimportant region. This means that disks can withstand much more violent shocks when they are powered down, but they are still quite delicate devices and must be treated gently.

Because disk drives rely on moving parts, they do not have the almost infallible nature and unlimited lifetimes of semiconductor circuitry. The minimal amounts of damage caused by vibration, poorly controlled power supplies, and just plain use (no moving part is completely frictionless,

so gradual erosion must occur) accumulate, until inevitably the whole drive fails. It must be understood that any data stored on disks is subject to unpredictable loss; the only solution is to keep backup copies on different media of anything that is essential.

The parameters of a disk that determine its capacity are the number of surfaces, the number of tracks, and the amount of data that can be packed into one track. Stating the number of tracks can be ambiguous (does 100 000 tracks mean 100 000 tracks per surface, or a total of 100 000 overall?). Rather than simply resolving the ambiguity with a decisive definition, a different term is usually used. A *Cylinder* is the stack of tracks at the same radial position on all surfaces, so a disk with five surfaces and 100 000 cylinders has an unambiguous total of 500 000 tracks overall.

On any single track, the data is not stored as one long stream of bits, but is divided up into a number of *Blocks*. A block is almost universally 512 bytes or 4096 bits. Until recently it was almost universal for there to be 63 blocks on each track, but now the number varies considerably, with the outer (longer) tracks holding hundreds of blocks. For efficiency of organization and access, and to ensure sufficiently accurate positioning, a block is the smallest amount of data that can be written onto a disk at one time. If a single byte is to be changed, it is necessary to read the whole 512 bytes that surround it, modify that one byte, and then write the entire 512 back again. In modern times, blocks are often erroneously called *Sectors*, but the term sector properly refers to the entire wedge-shaped collection of blocks at the same position on every track.

Disk drives are often referred to as *Hard disks* to distinguish them from *Floppy disks*, an alternative technology that through cheapness became very popular in the 1970s, but is now disappearing. Floppy disks are made of a flexible material and are much less finely constructed; early floppy disks (in the 8 and 5¼ inch formats) had less than rigid cases and were indeed slightly floppy. The read–write heads do not hover microscopically above the surface, but actually make close contact, pushing into and bending the surface. For this reason, floppy disks must rotate much more slowly than hard disks, have a much smaller capacity (generally only just over 1 MB), and are not kept spinning full-time. They also have a much more finite lifetime, but are correspondingly more robust, and can be dropped from a great height without harm. Floppy disks are encased in a square protective sleeve, which is also used to hold them in place when they are inserted into the drive. Floppy disks are always exchangeable: any number of different disks may be inserted into a single drive, used, then removed and replaced by another. Hard disks on small computers are never exchangeable, but the high-performance disk drives used on mainframes do still sometimes have an exchangeable disk pack as an option.

The first working disk drive in commercial production was the IBM 350, which was introduced in 1956 as a component of the IBM 305 computer. It was the size of a large wardrobe, had a capacity of 4.5 MB, and was available for lease at $35 000 per year. The first hard-disk drive available for an IBM PC, 26 years later, was the Seagate ST412, with a capacity of only 10 MB. IBM also introduced the first floppy disk, in 1971. In the 8 inch format, it had a capacity of approximately 0.1 MB, and was originally a system engineer's tool, not a general data-storage medium.

The *ZipDrive disk, Jaz disk*, and similar devices are two kinds of compromise disk technology. A lower-capacity version (around 100 MB) uses the same technology as floppy disks, but is much more finely and expensively constructed, so it gives higher capacity, access speeds, and reliability. The larger-capacity version uses the essential technology of hard disks, but with

exchangeable media in small cartridges that look somewhat like thick floppy disks.

Before a disk can be used in a computer system, it must be *Formatted*. This is the process of creating all the logical structures (such as empty directories or folders, lists of available blocks, and lists of file names) that the operating system expects to find. Formatting is an operating system dependent task: a disk formatted for one system may be completely unusable by another. It is also a repeatable task: a used disk may be reformatted to make it appear to be as new, so one can start again with a blank slate. Many operating systems provide two options for formatting: *Quick* or *Full*. A full format is usually a very slow procedure that often truly erases all old data on the disk. A quick format saves time by overwriting data only when necessary; old confidential data may be recoverable from a disk after a quick format.

When formatting a disk, most operating systems also provide the option of *Partitioning*. A *Disk partition* is simply the division of a large disk into a number of smaller regions, each of which is treated as a complete individual disk in its own right. This can have three advantages. Some operating systems have an in-built maximum disk size; partitioning an oversized disk can often reduce it to a usable size. Also, catastrophic disk failures, such as a head crash in an essential system directory, can easily affect a whole partition, but other partitions on the same disk have a greater chance of surviving unscathed. Finally, it can greatly simplify system maintenance to have multiple disk drives. If there is one disk for "system files" (the operating system and its data), another for applications, and yet another for user data, system upgrades can be much easier, since the whole operating system can be erased and re-installed without interfering with applications or data. Similarly, whole applications may be replaced without any risk to essential data. Having three disks adds significantly to price and power consumption; having one disk with three partitions adds nothing but convenience.

Business value proposition

Disks form the basis of secondary storage in all computers and the amount and type of storage that a corporate system requires are determined through the information lifecycle (see *Information lifecycle management*). Some small modern computing devices such as PDAs and true *Thin clients* do not have disk storage, and rely on non-volatile solid-state (*Flash*) memory or network access to disks attached to another system.

The floppy disk as a storage medium has shrunk in size since the 1970s, evolving from 8 to $5\frac{1}{4}$ to $3\frac{1}{2}$ inch, while the storage capacity has grown; however, their evolution seems to be coming to the end of the line as they are replaced by memory devices with no moving parts that can be connected to USB ports (flash memory). These devices have a significantly larger memory, and are smaller, more portable, and far more robust and reliable than floppy disks. As this evolution continues, organizations would be well advised to migrate to the newer alternatives. Flash memory does not yet approach the capacity of traditional hard disks, and continues to be many times more expensive per megabyte. The ability to add external and even removable hard-disk drives remains a useful option for network and system designers.

The price–performance ratio of disk-based secondary storage devices continues to improve for consumers; however, they should not be considered infallible and provision must be made by the network manager (and the individual) for the inevitable disk crash. While modern disk-based systems are more reliable than previous generations, they contain moving parts and hence will fail in due course. Systems managers need to have a device-replacement strategy built into their maintenance and capacity planning strategies.

Summary of positive issues

The cost of secondary storage in the form of disks continues to decrease while performance parameters increase. Each generation of disk-based memory is more robust and reliable than previous ones. Flash-memory devices are replacing removable floppy-disk drives and enabling users to have access to larger, more convenient, and reliable portable data storage. External disk drives are available to allow users of desktop and laptop computer to extend their storage capabilities.

Summary of potentially negative issues

The hard drives in computers do crash and it is imperative that a contingency plan should be put in place to backup and secure data. Hard drives in laptop computers are by no means insensitive to movement and can crash should sudden movements of the machine occur, especially if it is carried around while active, or if a laptop computer is knocked off a desk; contingency planning is thus also vital in these situations.

Associated terminology: Information lifecycle management, Backup, Optical storage.

Distributed database

Foundation concepts: Database, Parallel processing.
Definition: A database that is spread over a number of computers, distributing the load of storage and query processing.

Overview

One of the most important applications of *Distributed* or *Parallel processing* is to database systems. Corporate databases can quickly become very large, and may need to support multiple concurrent accesses from any number of branch locations. In many cases there is no super-computer that is super enough to handle the required load, so multiple database servers must be used instead.

Designing a distributed database is a complex task, and minutely tests the database administrator's and CIO's skills. Apart from the obvious problems of selecting suitable hardware, ensuring that it is up and running close to 100% of the time, providing secure and regular backups of data, and ensuring constant network accessibility, there are some fundamental technical questions of distributed design. The most fundamental of these is exactly how the database should *be* distributed; this is known as the problem of *partitioning the database*.

Should the data be divided up so that different tables (complete collections of data items of the same kind) are stored on different computers, which does not help when there is one major table that is both large and frequently accessed? Should the tables be *partitioned vertically*, meaning that each server holds a part of every record, or should they be partitioned *horizontally*, meaning that each server contains just some of the records, but each record on each server is complete, or should a combination of vertical and horizontal partitioning be used? Or should entire copies of the whole database be stored on each of a number of servers?

Whether and how a database is partitioned has a major effect on its operations. If multiple copies are kept on different servers, it is much easier to spread the load of query processing, since any query could equally well be answered by any server. It also becomes critical to keep the different copies of the database properly *synchronized*, so that, when a change is made to a record, there is no period of inconsistency, during which a query will produce different answers depending upon which server answers it. An inconsistent database can have disastrous results. Copying the database onto multiple servers also does nothing to reduce the problem of a

database being too large for one server to handle.

Horizontal partitioning does relieve the problem of too-large database tables. A database of 100 000 000 records could be split amongst ten servers, each holding 10 000 000 complete records. Any query will probably have to be sent to all ten servers, and the individual partial results will then need to be combined before being returned to the user, but that is one of the easier tasks. Horizontal partitioning usually implies the three-tier model of application servers, and often does little to relieve the problem of having too many queries to answer quickly, since each query must be processed by multiple servers.

Vertical partitioning may help to relieve both problems. With a database of 100 000 000 records, each of ten servers would hold the more closely related parts of every one of the 100 000 000 records. With careful planning, it can be arranged that the most common queries can be answered by a single server, but for those uncommon queries based on fields that never appear together on any one server, processing can become exceptionally complex and inefficient. With vertical partitioning, the problems of temporary inconsistencies immediately following insertions and deletions are greatly magnified.

The decisions on how the data should be distributed across multiple servers and the design of the algorithms for answering the various possible queries go hand-in-hand. One can not effectively be done before the other. The design of a distributed database is one of the most complex challenges facing an IT professional.

Business value proposition

As corporations grow, their databases grow with them. It is not unknown for some organizations to have databases of multiple terabytes, and sources have reported that some government agencies have databases of over 100 terabytes. Before a large database system can be designed, its general modus operandi must be decided: is it going to keep all of the data in a "live" transactional database that is immediately accessible, or is the data going to be pre-partitioned, with old data that is no longer likely to be accessed by online systems being placed in a *Data warehouse*, and only the live data remaining in the transactional database?

When an organization's overall requirements have been defined, then the database system can be designed. The design of a database is a technically challenging task and for any non-trivial case it requires a highly skilled and trained database administrator (DBA). The DBA needs to be skilled not only in database technology but also in the specific environment in which they are to operate. Databases differ from vendor to vendor, and from version to version, so very specific knowledge is needed.

Central to the DBA's task is the determination of how and whether to partition the data across a set of computers. There are several methods of partitioning, and the choice of solution is complex, involving a careful examination of the options. This requires knowledge of the interactions that occur between the hardware, the operating system, the network, and the application software, as well as understanding the practical constraints surrounding the database system itself.

The overall design of the database needs to be carefully considered, and the trade-off amongst performance, ability to scale, and cost must be built into the technical business case developed to help identify a solution.

Summary of positive issues

Distributed databases and partitioning techniques allow for a variety of solutions to managing very large databases. Distributed databases allow DBAs to attempt to optimize their database systems.

Summary of potentially negative issues

Creating distributed databases is extremely challenging technically. A poorly designed partition of the database can cause database failures, instability, and unacceptably poor performance.

Reference

- M. Ozsu and P. Valduriez (1999). *Principles of Distributed Database Systems* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Database administrator, CIO.

Domain name

Foundation concept: Internet protocol, Network.
Definition: A readable and memorable name associated with an IP address, providing a convenient reference to internet-connected computers.

Overview

All computers connected to the internet must have a numeric IP address associated with them. This is the only way that communications are routed across the internet, so it is an absolute requirement. IP addresses currently consist of four smallish numbers (range 0-255) separated by dots, for example "192.168.45.234." These numbers are not at all memorable, and are an extremely unsatisfactory way to publicize the address of a web site. With the advent of IPv6, these addresses will become even longer sequences of numbers (see *Internet protocol* for details).

A *Domain name* is a more user-friendly, human-oriented name associated with an IP address, having a more familiar and memorable form such as "MyCompanyName.com" or "AnotherCompany.co.jp." Before a domain name can be used, it must be registered with the internet authorities; there are numerous organizations providing domain name registration services for a fee; ultimate responsibility for domain names rests with ICANN (the Internet Corporation for Assigned Names and Numbers).

Once a domain name has been assigned to an organization, that organization is free to create any number of sub-names to refer to individual servers or internal networks, by prefixing extra components to the beginning of the domain name. For example, the owner of "company.com" has complete control over all names that end with ".company.com," and can freely define "www.company.com," "sales.texas.company.com," and anything else of a similar nature.

When a web browser, or any other internet client, attempts to access a server using a name like "sales.texas.company.com," a system known as DNS (*Domain Name Service*) is invoked. This is a network of servers, embedded in the internet, which provides the necessary information for converting names back to the numeric IP addresses that are required for internet communications.

There is a "*Top Level DNS Server*" for all names ending in ".com"; this server is the ultimate authority for all ".com" addresses, which is why proper domain name registration is essential. The top-level server is asked what it knows about "sales.texas.company.com," and responds with a partial answer: the address of another DNS server that is responsible for all addresses ending in ".company.com." The question is then asked again of this server, and it may reply with the complete answer, or it may instead reply with the address of another DNS server that is responsible for all names in the ".texas.company.com" subdomain. Eventually, if everything has been set up correctly, the question will reach a DNS server that knows the complete address, and communications may begin.

When a domain name is registered, it is also necessary to make sure that there is at least one DNS server that can answer DNS queries about hosts within the domain.

For an organization that does not have its own 24-hour computing service, this will probably involve additional fees. It is normally considered essential to have two DNS servers prepared to handle queries about any domain, since some down-time is inevitable, and without a working DNS, internet sites can not be found.

The last component of a domain name does have some significance, and can not be chosen freely. There are seven traditional top-level domains: ".com" was intended for multinational companies, but has come to be used by US companies; ".edu" is for US educational institutions; ".gov" is for US government agencies; ".int" is for international organizations; ".mil" is for the US military, ".net" is for networks, and ".org" is for other US organizations. In order to relieve demand on the exceptionally popular ".com" names, six new ones were created: ".biz," ".info," ".name," ".aero," ".coop," and ".museum." Non-US individuals and organizations are expected to use country-specific domain names; these end in two dotted components, one indicating the kind of organization, and the other (always two letters) indicating the country it is based in; so, for example, ".co.uk" is the UK-specific version of ".com," and ".co.jp" is the Japanese. There is a ".us" code for the United States, but it is rare for a US corporation to use ".co.us" instead of ".com."

Business value proposition

The choice of a domain name is a vital aspect of corporate branding, for large established corporations their first choice of domain name was most likely secured within the first few years of the internet being deregulated to allow commercial exploitation. There have been many instances of individuals *Cybersquatting*: buying up the domain names associated with famous brands and then attempting to extort the legal holders of the brand name. To prevent this from happening, the United States passed the Federal Anti-Cybersquatting Consumer Protection act of 1999. However, for smaller organizations or individuals, obtaining a "meaningful" domain name can be problematic, since many popular names and terms have been used already, e.g., www.robertplant.com. This can cause the need for a re-branding effort.

The choice of domain name of course does not have to be limited to local names, e.g., a company in the UK can choose to have a .com address, or even a co.us address. It is, of course, very difficult for people to guess a domain name and even powerful search engines might not easily help a potential customer locate the company they are looking for without specific information. There is no domain-name-index equivalent of the "Yellow Pages," so for example, finding a hardware shop in London that has not had much internet traffic or been in business long could be difficult.

Summary of positive issues

The domain name-space is regulated by ICANN. Domain names remove the need for people having to remember IP addresses. Domain names are unique. The configuration of a DNS server is relatively easy and connections through ISPs are available almost universally.

Summary of potentially negative issues

Domain names for popular words, phrases, and names are frequently already registered. Top-level domains such as .biz and .name are not popular and not widely recognized by the public.

References

- http:/www.icann.org.
- ICANN, 4676 Admiralty Way, Suite 330, Marina del Rey, CA 90292–6601, USA.

Associated terminology: ISP, DHCP, Client–server.

Dvorak/QWERTY keyboards

Definition: QWERTY is the standard layout of letter keys on keyboards; Dvorak is an alternative layout.

Overview

The strange rendition of the alphabet found on typewriters and other keyboards, Q W E R T Y U I O P A S D F G H J K L Z X C V B N M, is neither random nor arbitrary. It was deliberately designed (by Christopher Sholes, around 1870) with two ends in mind. One was to reduce key jams by slowing down the rate at which common letter sequences could be typed by placing the constituent letters far apart. The other was to ensure that the letters of the word *typewriter* were all on the top row, as a sales gimmick. It is clear that there must have been other design criteria that were not recorded in the annals of history.

Now that mechanical typewriters are things of the past, key jams are no longer a concern, and the limited benefits of the QWERTY layout are outweighed by its disadvantages. The QWERTY layout is far from optimal: it requires frequent uncomfortable movements that do slow down typing, and are believed to contribute to repetitive-stress injuries.

The Dvorak (often spelled Dvorjak) keyboard was designed by August Dvořák to optimize the key layout for fast and comfortable typing. He called his layout ASK (the American Simplified Keyboard), but his own more interesting name is always used instead. The main design features are that the most commonly used letters all appear on the middle row, the most commonly used sequences involve alternating hands and progress from little fingers to larger fingers (a natural motion for humans), commonly typed sequences involve minimal movement of the hands and reaching with the fingers, and the load of keypresses is shared more equitably between the fingers, slightly favoring the stronger ones.

Business value proposition

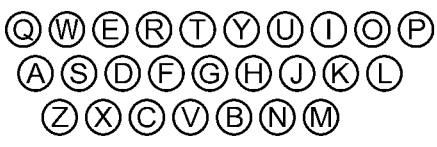
The selection of a keyboard is usually made by default, without any thought, by the vast majority of businesses or individuals; they simply take what comes with the computer from the manufacturer. However, the long-term productivity perspective could suggest considering a change to a Dvorak or an even more ergonomic keyboard. Several manufacturers produce ergonomic keyboards that are more comfortable, with lower key-impact requirements, separate keyboards for each hand, and adjustable key spacing. While rapid change is unlikely, the advent of cellular telephone text messaging, which is based on alphabetical key ordering, may result in future consumers being more open to changing keyboard formats.

Summary of positive issues

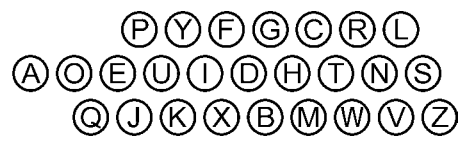
Tests do reveal that learning to type is easier and faster with the Dvorak keyboard, and that, once the keyboard has been mastered, typing is faster and less stressful.

Summary of potentially negative issues

Dvorak keyboards are not very commonly used, and so are not produced in very large quantities, and as a result are more



(The QWERTY keyboard)



(The Dvorak keyboard)

expensive than the QWERTY standard keyboards. Typists need to be retrained (or at least to retrain themselves), and often resist the change. Switching to Dvorak keyboards incurs some expense, and a period of much reduced productivity while typists gain familiarity with them.

Reference

- R. Soukoreff and I. MacKenzie (1995). "Theoretical upper and lower bounds on typing speed using a stylus and soft keyboard," *Behaviour & Information Technology*, **14**, 370–379.

Dynamic web pages

Foundation concepts: Web page, Internet, HTML.
Definition: Web-page content that is not static, but can vary according to circumstances, and react to user inputs and requirements.

Overview

The most basic web pages are simple text files with a fixed HTML content. Every time a particular web page is accessed by a browser, the content shown will be exactly the same. This is, of course, perfect for providing fixed information that does not vary, but provides only limited possibilities for user control and interaction (no more than the ability to click on a link to another page), and makes truly interactive operations such as online registration, form submission, hit counters, shopping carts, and database queries impossible.

Dynamic web pages provide content that is not created until it is needed; with each browser access, the page is created on-the-fly, and thus may be different each time and can even react to browser input. There are five very commonly used technologies for dynamic web page creation: SHTML, CGI, JSP and ASP, and Scripting, and many others that are just variations on a theme.

To understand dynamic web page generation, it is essential to understand the basic nature of web interactions. When a web browser wishes to access a particular page, it composes a simple command in the HTTP language. This command usually begins with the word "GET," and contains an identifier for the desired page (its URL), together with any user inputs that may have been provided in a form. This request is transmitted to the relevant server. When a web server receives a GET request, it translates the URL into a simple file name, and looks for that file. If the file contains simple HTML or plain text, the entire file is sent, unprocessed, to the requesting browser, which then displays the content on the screen.

If the server finds that the file actually contains *Server-parsed HTML* (SHTML), then the processing is slightly more complex. Instead of just transmitting the entire file as-is, the server reads the file looking for special tags. Most of the file is simply sent to the browser as-is, but *Server-side includes*, surrounded by "`<!--#`" and "`-->`" are replaced by the appropriate text. For example, if the text "`The file xxx.html is <!-- fsize file = "xxx.html"-- > bytes long`" were to appear in an SHTML document, and the server had a file called `xxx.html` 7055 bytes in length, then the text "`The file xxx.html is 7055 bytes long`" would be sent to the browser. SHTML is of only limited power, but does increase the scope of simple web pages.

If the requested file is of the CGI (*Common Gateway Interface*) type, then the server assumes that it is an application program, and executes it. The running program receives any inputs that were sent as part of the request, and all output created by the running program is sent directly to the requesting browser. Anything that the server is capable of doing can be programmed into a CGI application; using CGI is just like running an application over the web. Programmers can use any language that the server supports, and have

access to all operating-system services. CGI is by far the most powerful and flexible of the dynamic generation methods, and that does have its drawbacks. Writing CGI code requires real programming knowledge, and is not a task for unskilled workers. There is also a security risk: CGI can do anything, so a careless programmer could accidentally open up the whole system to outside attack. In skilled hands, CGI is safe, simple, and powerful.

JSP (*Java Server Pages*) and ASP (*Active Server Pages*) are very similar technologies. Both are based on the idea of SHTML: the server processes the file as it is being sent to the browser, and substitutes information generated on-the-fly for special tags in the text. With both ASP and JSP, the special tags contain parts of (or even complete) programs that are executed to generate the new content. These program segments can do anything from providing the current date and time to substituting the results of complex cross-network database queries. In JSP the language of the program elements is based on Sun's Java; in ASP it is based on Microsoft's Visual Basic. The use of JSP requires Java support on the server; the use of ASP requires a Microsoft server.

Scripting is a different kind of system. When scripting is used, the file retrieved by the server contains program code, but is not processed by the server at all. The file, together with any code it contains, is sent directly back to the requesting browser, and the program code is run locally by the browser. This has the positive effect of relieving the server of a lot of the processing load, but at a very high price. It is the browser that has to execute the script, so it has no access to any information stored on the server. Of course, scripts may open communications channels back to the server, but that adds a great deal of complexity both to the script and to the server, and somewhat defeats the purpose of scripting. Scripting elements are commonly used to make the cursor or hypertext links change their form depending on the position of the mouse. Scripts are most commonly written in JavaScript and Visual Basic.

Business value proposition

Dynamic web pages created through the SHTML, CGI, JSP, and ASP web technologies allow businesses to build more sophisticated web sites that facilitate a variety of activities ranging from gaming to B2C e-commerce and B2B data exchanges. The web technologies allow adoption of a variety of processing models; for example, in the scripting model the processing is at the client, whereas the JSP model performs the processing at the server, and this flexibility allows developers to create systems that are matched to their business need and their systems architecture and to take into account the technologies deployed by their customers, partners, and vendors.

Summary of positive issues

Dynamic web content is necessary if web sites are to go beyond merely providing static information, and interact with users. e-Commerce can not be conducted with merely static web pages; the closest approach would be to provide customers with forms that they must print, complete by hand, and fax in to make an order, and such a procedure would certainly discourage many potential customers.

CGI has unlimited power, allowing anything a programmer is capable of coding, but is lacking in convenience, and does require some technical programming ability. Javascript provides much greater protection against both accidents and malice, but has a much greater likelihood of arousing the suspicions of a user's security system and being blocked. Scripting allows simple user interactions without any load on the server.

Summary of potentially negative issues

Dynamically created pages put an extra load on the server, the browser, or both. Careful capacity planning is required to ensure against systems being overwhelmed at times of peak demand.

CGI and, to lesser extents, JSP and ASP require a degree of technical programming ability from the originating organization. CGI requires only very basic support and setup on a web server, but JSP and ASP require more significant administration, and generally put a heavier load on a server. CGI's power means that careless programmers can do unlimited damage; CIOs must be sure of the competence and loyalty of any programmers selected to work on server and CGI implementations.

The more sophisticated dynamic web pages become, the more likely they are to arouse the suspicions of client-side security systems (particularly firewalls, web-browser security settings, and anti-virus software). Content that gets blocked, even if it is completely innocent and harmless, can have a negative impact on the corporate image in the customers' minds.

Associated terminology: Javascript.

e-Commerce/e-business

Definition: e-Commerce/e-business are business models that utilize the internet as a mechanism through which companies, vendors, and customers interact.

Overview

e-Commerce originated with the deregulation of the internet in 1995. It was originally conceived as a business-to-customer (B2C) model of commerce, and corporations were formed to provide services and products over the internet without having any physical retail presence (e.g., Amazon.com, the well-known online book seller).

The term *e-Business* was coined by Lou Gerstner, then CEO of IBM, who looked beyond the B2C model to use the internet for delivering services and supporting physical business channels.

The start-up companies that adopted pure B2C commerce, for which there is no physical retail store, together with start-up business-to-business (B2B) organizations, which also owned no physical retailing or warehousing capacity, aimed to "disintermediate" the established companies by having lower internal and distribution-channel costs. Unfortunately, this model proved to be very expensive to support as the businesses attempted to scale up and reach profitability, and the majority ran out of capital following the decline in the US stock market of 2000 and closed down. Those that survived the stock-market crash endured a market consolidation and continue to attempt to build market share, drive down costs, and become profitable or maintain profitability.

Traditional companies with physical retail and supply-chain facilities (also termed "clicks-and-bricks" companies) have continued to develop their models and focus upon those aspects of the business model that provide profitability (e.g., Walmart.com).

The B2B business models have also evolved as essential elements of the internet infrastructure as HTML and XML have become more established. Several models have become well established, including data-synchronization hubs, e-procurement industry-specific systems, and specialist e-procurement systems. Data synchronization hubs such as Transora.com provide the members of that hub with the ability to synchronize data with other members and notify specific members of events such as price changes or product-specification changes. Industry-specific e-procurement systems provide members with the ability to perform a range of procurement activities through a central hub; a key example is COVISINT, which provides procurement services for the automobile industry. Specialist e-procurement systems provide specialized or niche services, such as Chile Compra, which provides procurement services for the Chilean government (https://www.chilecompra.cl/).

Business value proposition

B2C models can provide businesses with an alternative retail channel if they are developed in alignment with the overall corporate business strategy. Four factors can be considered to help "position" a company's B2C online presence: brand, service, market, and technology. The brand strategy could be to reinforce an existing brand, create a new brand, or use the internet to reposition a brand. All of these strategies require that brand equity modeling be used in order to determine the most appropriate branding strategy to utilize. The online brand that an organization attempts to project is driven by three other positional factors: *service* levels must match the brand requirements, the *market* being addressed must offer products that are aligned with the company's desired brand position, and the *technology* must be adequate to support these conditions.

B2B models provide businesses with the potential both to lower transaction costs and to increase the speed of transaction

data exchange. Data-synchronization B2B systems can be in the form of consortia, with the members benefiting from lowered development costs since they share the overhead associated with technical development. e-Procurement portals also allow members to reduce overall costs by avoiding the search effort involved in accessing multiple corporate sites, hosting a site themselves, or competing against each other with proprietary systems.

Summary of positive issues

Many successful B2C models have emerged, and traditional companies such as the UK supermarket chain Tesco (www.tesco.com) understand the importance of building upon established strengths and ensuring that the online business model is aligned to the overall corporate strategy. Tesco's business model is such that physical orders taken over the web (e.g., groceries, clothes, DVDs) are filled and delivered from the store nearest to the customer, while service products (e.g., car insurance, holidays, and flights) are provided via the web.

The technical costs associated with B2C and B2B systems developments have declined as tools, standards, support, and methodologies have been evolved both by commercial vendors and by organizations such as the World Wide Web Consortium (W3C.org).

B2B models such as data synchronization centers provide companies with the ability to share and distribute data to a large number of businesses and entities at a lower cost than would be possible with a proprietary system. Membership of a B2B procurement exchange reduces the transaction costs associated with the procurement process and reduces costs for members because they do not have to build the procurement system themselves.

Summary of potentially negative issues

B2C models continue to evolve as emerging technologies change the customers' ability to locate and use alternative sources of supply. Organizations need to evolve their sites accordingly. Lack of alignment in corporate branding, service, or product offerings for bricks-and-clicks organizations can lead to the loss of customers or confusion on the part of customers as to their relationship with the company.

Data-synchronization B2B systems require a relatively high degree of technical, process, and organizational sophistication of active participants. B2B procurement hubs may evolve until a particular hub becomes dominant; selection of hub membership is an important strategic sourcing issue.

Reference

- R. Plant (2000). *e-Commerce: Formulation of Strategy* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Internet, Dynamic web pages, W3C.

Efficiency

Foundation concept: Algorithm.
Definition: A measure of how well resources are used by an application.

Overview

The general scientific definition of efficiency is the output actually produced by a system expressed as a fraction of the maximum output any such system could theoretically produce. An engine is 50% efficient if half of the energy put in (in the form of fuel) is converted into useful work (the other half being lost through friction, pollution, etc.)

While this definition can be useful in computing, it is not the definition normally used by computing personnel. In most cases there is no "theoretical best" performance that an application may be measured against. In computing, the term

"efficiency" is usually used as an umbrella term for absolute measures of performance.

For example, it may be necessary to process a large amount of information to produce a summarized report. Application A might load all of the information from disk into main memory before processing and thereby produce faster results than application B; which might perform exactly the same task only taking the data in smaller chunks, and therefore working much more slowly. A naive observer would conclude that application A is better because it produces the same results but faster. A deeper analysis is required.

For example, tests may be performed, comparing the two applications on a number of different data sets:

Data-set size	Time for A	Memory for A	Time for B	Memory for B
20 MB	0.15 s	21 MB	6.00 s	2 MB
40 MB	0.30 s	41 MB	12.00 s	2 MB
60 MB	0.45 s	61 MB	18.00 s	2 MB

From this (trivially simple) experiment, one might conclude that application A requires 0.0075 seconds per megabyte of data, whereas application B requires 0.3 seconds per megabyte, so application A is 40 times faster than application B. But the amount of memory occupied by application A grows alarmingly as the size of the data set grows, whereas application B uses only a constant amount of memory. Of course, there are many other factors to be taken into consideration, such as how much human intervention each application needs, how much network bandwidth they require, and how much memory the application processing the data requires.

This illustrates an important concept: there is often a trade-off between one form of efficiency and another. To make a program run faster, it may need more memory, but to make a program run with less memory, it may have to be made slower.

There is of course a fundamental difference between time efficiency and memory efficiency. Fast execution is usually a most desirable quality, but, if a program is slow, all you have to do is wait a little longer and the results will appear. If a program needs too much memory, there is nothing to be done about it.

The efficiency of human effort is another factor that must not be forgotten. If a programmer spends a week making one part of an application more efficient, so that it now takes one second to run instead of ten, is that an effective use of resources? The answer of course depends on circumstances. If this part of the application is run many times by customers, then the total improvement can be enormous.

Business value proposition

The methods used to determine the efficiency of normal engineering problems typically allow clear results to be determined. Computing systems do not always provide such clear-cut decisions. In reality there are trade-offs between the different ways of solving any given computing problem; for example, using more processing power against using more memory or vice versa.

A programmer who is aware of the constraints applicable to the system being developed can attempt to construct applications that maximize some quantifiable measure of efficiency. The determination of efficiency within large corporate systems environments is a very difficult problem to solve, due to the interaction of many systems-related variables. For example, a corporate web server servicing simple web-viewing requests involves many processes and sub-processes, including the network-data-traffic profile, the efficiency of the web-page design, the hardware that supports the system, and the efficiency of the

firewall. Each factor has to be taken into account when determining the efficiency of the overall system. Typically, stress and load tests are undertaken to determine whether a system will support the average and maximum loads to be placed upon it.

Many applications that are highly sensitive in terms of their efficiency requirements are carefully examined, and, when deployed, reside upon their own platforms in order to maintain the integrity of those performance characteristics.

Summary of positive issues

The theoretical efficiency of a critical strategic system can be determined. This is vital for systems whose failure to perform at a certain level may involve serious problems with processes or, more importantly, risk to human life, e.g., aircraft systems. Stress and load tests can be used to determine performance and efficiency profiles empirically.

Summary of potentially negative issues

The determination of systems efficiency is a complex and time-consuming problem and failure to do it for a strategic or critical system can be extremely detrimental to an organization and its performance. It is essential to decide exactly which aspects of efficiency are most important because one aspect might be increased only at the expense of another decreasing.

References

- N. Wirth (1978). *Algorithms plus Data Structures Equals Programs* (Englewood Cliffs, NJ, Prentice-Hall).
- D.E. Knuth (1975). *Fundamental Algorithms* (New York, Addison-Wesley).
- M. Hofi (1975). *Analysis of Algorithms* (Oxford, Oxford University Press).

Associated terminology: Computability, Complexity.

Electronic data interchange (EDI)

Foundation concepts: Network, Database.
Definition: Electronic data interchange is the transmission of data or signals between two or more entities.

Overview

The origins of electronic data interchange (EDI) go back as far as the beginning of commercial computing (or data processing as it was commonly referred to in the 1960s and 1970s). EDI exists to connect two organizations that need to transfer or exchange information.

In early EDI, data-exchange methods were created in an ad hoc manner to solve individual problems between, for example, a manufacturer and a vendor, and scalability and compatibility of technology were not seen as important. Typically a company would have many "standard" file structures, with different versions being created for each client, buyer, or supplier as needed.

Not only was managing the different file structures and data types problematic, with companies incurring large maintenance overheads, but also it had the side effect of locking companies together, trapped by the proprietary data formats they shared. This had two effects; for suppliers it created a major barrier to entry against competitors who would usually not have the same file-format standards, and for customers it resulted in switching costs each time they changed data format.

The proprietary advantages enjoyed by companies who forced their data formats upon their business partners could only last so long, and it became apparent to all parties, including the United States government, that this situation was unfair and anti-competitive. In January 1986 a federal mandate was issued, stipulating that certain industries, including healthcare, could not use proprietary EDI formats and that only one standard messaging system

would be allowed, that being defined by the American National Standards Institute and known as ASC (X.12). Several groups, including the United Nations and member countries, have carried the development of this data-exchange standard forward: UN/EDIFACT (United Nations Rules for Electronic Data Interchange for Administration, Commerce and Transport), The European Board for EDI Standardization (EBES) through the European Committee for Standardization (CEN), and the Japan Association for Simplification of International Trade Procedures (JASTPRO) created a series of EDI protocols and associated standards.

The rise in importance of electronic data interchange and the need to pass different data types between entities have caused the term EDI to be refined, with EDI being used to refer to data-processing-type transactions and inter-organizational systems (IOS), and "e-business" referring to the transfer of more flexible data types between entities.

IOS has been facilitated by the rise of open standards such as the Extensible Markup Language, known as XML, which is a simple, flexible text format that originated from the ISO 8879 standard and the advance of network protocols such as TCP/IP.

Business value proposition

EDI has evolved to allow highly flexible messaging between two or more entities. Two main approaches are followed: the exchange of fixed file formats such as those found in the X.12 standard created by ANSI, and the exchange of flexible file formats through the Extensible Markup Language (XML) which has been developed under the auspices of the World Wide Web consortium (W3C.org) and the W3C XML working groups. Both approaches allow entities to connect to each other and send data in an efficient manner.

Summary of positive issues

EDI is used to send millions of messages and files every day all over the globe; both the X.12 and the XML approach are widely used and are undergoing constant revision. The X.12 standard was created in 1979 and is widely used, with a large body of industry knowledge and support to assist X.12 users and developers. The standard covers a very wide range of documents, with approximately 300 transaction sets having been developed. The X.12 standard from Version 4010 onward is year-2000 compliant and the US federal government mandated that healthcare providers be X.12 compliant by August 1, 2003 in accordance with the 1996 Health Insurance Portability and Accountability Act (HIPAA). XML, a platform-independent approach to data exchange, also continues to grow in popularity, with both open-source and proprietary software available to support its user base. The use of X.12 and XML provides companies with the opportunity to reduce transaction costs, speed data exchange, and simplify procedures.

Summary of potentially negative issues

The dual approaches open to organizations wishing to undertake EDI (X.12 and XML) for data messaging can cause an increase in overhead if both mechanisms have to be employed. The technical and human-resources overhead associated with implementing an X.12 solution can be considerable and it is less flexible than an XML solution. However, X.12 is evolving toward the incorporation of an XML component. XML adoption by companies and software vendors continues to increase, due in part to its flexibility. As an open standard, XML may be amended and changed to accommodate new requirements faster than other formats.

References

- http://www.x12.org/x12org/international/index.html.

- http://www.unece.org/trade/untdid/.
- http://www.cenorm.be/default.htm.

Associated terminology: XML, X.12, Health Insurance Portability and Accountability Act.

Email

Foundation concepts: Internet, Protocol.
Definition: A system for exchanging messages and data in a non-interactive mode.

Overview

Email has been around, in one form or another, for about 40 years. Originally, most computer users had an account on a mainframe, a single large shared computer. If one user wanted to leave a message for another, they could simply store some text in a particular disk location associated with that user's account, and the next time the intended recipient logged in they would be shown the message. Very simple software automated the tasks of sending and receiving email, and the resultant system was popular, low-maintenance, and easy to use.

The coming of the internet complicated matters, because users now expected to be able to send email messages to correspondents on remote computers. This was not a particularly difficult enhancement to the system; all it needs is a special-purpose application running permanently in the background on each email-enabled computer, waiting for email messages to arrive over the network connection, and dealing with them exactly as local emails had been dealt with before. The email-sending application simply had to know how to transmit outgoing messages.

What complicated email was the coming of the personal computer. Personal computers spend a lot of time turned off or otherwise disconnected from the internet. Email that arrives when a computer is turned off must still be accepted and kept somewhere, and delivered as soon as the computer is online again. That is why email servers came into existence, together with the internet email protocols SMTP, POP2, POP3, and IMAP.

To receive email, a person must have an account on an *Email server*. Any computer can be an email server, the required software is widely available and puts little load on the system. The only requirement for an email server is that it must be permanently on, and permanently connected to the internet. To be useful, it must also have a *Static IP Address*, but that is not strictly necessary. Having an account is not an onerous requirement; it simply means that the server must know about you (you have a user-name and a secret password) and it must set aside a little disk space to keep your messages until you read them. ISPs usually provide their customers with access to an email server as part of the package.

When email is sent to a remote computer, an internet connection using TCP/IP is made to the SMTP (Simple Mail Transfer Protocol) service application that must be running on that computer. After a short exchange in which the sender and recipient are identified, the message itself is transmitted and the connection is broken. An SMTP server keeps a special file or folder, known as a mail box, for each known user; the new message is appended to the end of the appropriate mail box.

Whenever an email-browser client is started, it automatically makes an internet connection to any email servers that it has an account on. This time, it does not connect to the SMTP server, which is used only for sending, but connects to the POP2 (Post Office Protocol version 2) server, the POP3 server, or the IMAP (Internet Message Access Protocol) server. All three of these provide the same basic services. IMAP is the most sophisticated, providing many additional features; POP2 is the simplest.

Users are normally able to select which is to be used in the options for their email-browsing client.

When the email-browsing client successfully connects to the POP or IMAP server, it identifies itself by sending the user-name and secret password, and is given a list of any unread emails waiting for it. Usually, it will then ask for each of those messages in turn, and request that they be deleted, then display the list of messages to the user, and allow him or her to do with them as they will. The whole procedure of connecting to the POP or IMAP server and retrieving the list of new messages is repeated every few minutes for as long as the browser continues to run. That is the only way that new messages are detected; the server has to be asked explicitly, it will never send a notification.

Email messages were originally nothing more than plain text, but modern systems allow any kind of data to be attached to the message. Attachments are sent to the SMTP server as a continuation of the main message, all in one unbroken transmission. It is the email-browsing client that picks out the attachments and treats them as separate entities. Modern systems allow whole web pages to be sent as the main message, so it is displayed to the recipient with all typesetting and images in place. If the email viewer has insufficient security settings, such a message may also contain an embedded program that will be run automatically as the message is displayed. This is a sufficient opening for a virus to be installed and activated.

Broadcasting is not possible with current email systems: every message must have a specific recipient stated, so sending to multiple recipients requires multiple sendings. There are products known as ListServs (no error: there is no "e" after the "v") that automate the sending of a single message to a long list of multiple recipients. Email clients generally understand the idea of "mailing lists" and will automatically send a message to all members of a predefined list.

Business value proposition

Email has become one of the most valuable of all corporate communication tools. Email is accessible almost universally on a wide variety of devices, and has become almost indispensable in today's organization.

The technology of email has evolved sufficiently that end users are unaware of the majority of technical issues involved, since these fall upon the network manager. Users do, however, expect a seamless service from their systems and demand as close to 100% up-time as possible, and, when the "email server" is down, they expect all unread emails to be available when the system comes back on line. While the creation of an email server is quite straightforward, and in an ideal world providing 100% reliability would be no problem, this is not the case in a real-world implementation.

The email systems of organizations are in essence the digital equivalent of a corporate "front door" and everyone's email comes and goes through it. This makes it both vital from a strategic communications perspective and simultaneously a key point of attack for those with evil intent. The network manager may employ a variety of techniques to prevent email from being attacked and these include proxy servers and firewalls to prevent spam attacks and emails with dangerous attachments entering the system; Trojan horses, viruses, worms, and other programs that users unfortunately open by mistake are a primary concern. It is problematic that these malevolent system attacks continue to evolve and change, forcing the network manager and corporations (as well as individuals) to deploy ever-increasing resources to defend against them. One of the worst scenarios for network managers to consider is the complete loss of the email server with the discovery that there is

no backup. Many system managers have rightly lost their jobs over this neglect and it is vital that a backup procedure is in place and checked frequently to ensure that it works; even a few hours' worth of lost emails can be considerably damaging to many organizations. The network manager needs to consider issues such as capacity planning for the server and the memory demands placed upon it by users. Many users save all their emails and wish to access them all from their inbox and are reluctant to archive them, their email box acting as a "Rolodex." This can place significant storage demands upon the file server and the network manager will usually put in place quotas and other procedures to prevent users being wasteful of storage.

Many companies have strict policies about the type of emails that can be sent (both internally and externally), since they are documents of the organization and as such will be archived by the organization, and are subject to discovery in legal proceedings. Workers need to understand this basic point and must not send internal emails saying "don't buy this stock, my research shows it is awful" and then send out external emails to customers saying "buy this stock, it is great" because these conflicting positions may one day be used in a lawsuit. For legal and other reasons some people simply prefer not to use email. President G.W. Bush stated "I don't email," adding "and there's a reason. I don't want you reading my personal stuff." (speech at the American Society of Newspaper Editors conference in Washington, DC, April 14, 2005).

For most people email is too critical to day-to-day life for it to be neglected, but email can become very intrusive and detract from workers' productivity. Again procedures and policies need to be put in place to assist productivity. Executives are also frequently the subject of direct emails from customers and vendors, some legitimate, some not. It is advisable for executives to have private email addresses as well as their more public ones; the non-private emails can then be filtered through an assistant. For example, it is particularly easy for people to guess an executive's email addresses if everyone in the company has the same email address form (e.g., psmith@company.com, jsmith@company.com). Some system managers establish email address lists for prevalidated correspondents and filter incoming email for executives through that list.

Summary of positive issues

Email aids communication when used in a filtered, secure environment. Email allows users to access communication asynchronously (when they want to, rather than when the system or external forces decide it is time). Email servers are easy to establish and the technology protocols are well known. Third-party email systems are available and accessible via the internet. Email can carry attachments and can be sent out to many addresses at once via a ListServ. Email can provide an audit trail of correspondence and may be required to be archived for regulatory compliance. Email can be accessed through a wide variety of devices and services.

Summary of potentially negative issues

Email can facilitate attacks on corporate networks and individuals through spam and malicious attachments. Email can reduce productivity: the easier it is to ask a question, the more likely a person is to ask that question without thinking about it first; since the days when access to customer support required writing a letter or paying for a long-distance telephone call, the number of totally unnecessary enquiries has grown enormously. Employees will often spend an inordinate

amount of time sending and responding to personal emails if a proper policy is not enacted and enforced. Archiving requires the expenditure of resources by network managers to ensure regulatory compliance.

References

- D. Wood and M. Stone (1999). *Programming Internet Email* (Sebastopol, CA, O'Reilly Press).
- S. Cobb (2002). *Privacy for Business: Web Sites and Email* (Saint Augustine, FL, Dreva Hill LLC).

Associated terminology: Instant messaging.

Encryption

Foundation concept: Security.
Definition: Disguising information to make it inaccessible to certain others.

Overview

Obscurity is not security. Hiding information, or encoding it using a strange process that "nobody could ever guess," provides no security at all. One disgruntled employee, one inspired enemy, and all is lost. Since the first electronic computer was built in 1943 for the specific purpose of breaking enemy codes, data security has moved firmly into the realm of abstract mathematics. Anybody who tries making up their own encryption schemes without extensive knowledge of cryptanalysis is setting themselves up for disaster.

The simple encryption schemes that have been tried and tested for centuries, which are based on alphabetic substitution (for example, $A \rightarrow K$, $B \rightarrow F$, $C \rightarrow W$, $D \rightarrow A$, . . .) and permutation (for example, hello $\rightarrow$ lhloe), can be cracked in a fraction of a second with the aid of a computer. A new encryption scheme may seem safe to its inventor, but could crumble quickly under attack from unsuspected techniques. The only way to be confident that an encryption method is safe is to use one that is well known and has already withstood years of attack by the entire cryptanalysis community.

Encryption always depends upon a *Key*, something like a secret password or number that plays an essential part in the encryption process. The same encryption scheme used on the same message, but with a different key, will yield completely different results. The safety of encryption is based on keeping the key secret, not the method. The simplest of cracking techniques, and in many cases the most successful, simply involve trying out every possible key. If, for example, encryption keys are four-digit PINs, then there are only 10 000 possible different keys. A computer could try them all out in turn in a fraction of a second, and be guaranteed to crack the message.

It is always essential that there should be so many possible keys that they could not possibly all be tried. This is clearly illustrated by the example of *56-bit DES* (*Data Encryption Standard*), a well-known worldwide standard. The 56-bit DES encryption scheme supports 2^{56} (which is about 70 000 000 000 000 000) different keys. A 4 GHz Pentium 4 could probably perform about 7 000 000 experimental decryptions per second, which means that it would take around 300 years to try every possible key. That may sound secure, but it means that any industrial rival could simply buy 300 computers, leave them running for a year, and have everything. Or 3000 computers could do it in about a month. Deploying 3000 computers would cost some money, but there are economies in scale, and the value of being able to read all rivals' secrets would certainly be much higher. To make matters worse, special-purpose integrated circuits have been fabricated that are many times faster. DES is expected to be superseded by *AES* (*Advanced*

Encryption Standard); it has a more open design, and supports 256-bit keys. AES is the first encryption method approved by the US government for "top secret" documents to be released to the public.

Increasing the key size has an inordinate effect on the security of the system: 56-bit DES keys are approximately 16-digit numbers; 112-bit DES keys are approximately 32-digit numbers, a fairly small increase in size, but a major increase in security. It increases the time required to try all possible keys on a farm of 3000 computers from one month to 2 000 000 000 000 000 000 years. Of course, there is no practical limit to the size of a key, except that legitimate encryption and decryption by those who do know the key will also take a little longer.

The need for large key sizes brings about its own problems. Nobody can be expected to remember a 32-digit number, but if ever it is written down or recorded in any way, simple theft becomes an attractive means of breaking the best encryption. Key security and key exchange are serious problems for the security-conscious organization.

There are two basic forms of encryption method: symmetric, in which the same key is used for encryption and decryption; and asymmetric, in which case the key that must be used to read a message must be different from the one used to encode it. Symmetric systems require that a different key is produced for each possible pairing of communicants: if four people A, B, C, and D wish to communicate securely, they could pick a single key for their group, but that would mean that they have no internal privacy. Instead six different keys are needed, one for when A and B communicate, one for when A and C communicate, one for A and D, one for B and C, one for B and D, and one for C and D. To give N people private communication, approximately $\frac{1}{2}N^2$ keys are needed; the key-management problems can become substantial. With an asymmetric system, it is necessary to have just one key pair (an encryption key and a decryption key) for each individual involved, and all combinations of secure communication become possible. See *Public key–private key* for further information.

Another encryption scheme of value is the *One-way hash*. This is a kind of encryption that can not be decrypted even if the key is known. The original message is never recoverable even by legitimate readers. Although it sounds pointless, this technique is very valuable, and makes possible *Digital signatures* and *Digital certificates*.

All encryption schemes (with one exception, the *One-time pad*) are in principle crackable. The only security lies in the fact that the entire world's cryptanalysts are working on them and have been for a long time. It would be very difficult to keep a flaw secret, and what discoveries one lab does manage to keep secret will probably soon be repeated by others.

Business value proposition

There are many tried and tested encryption algorithms, and the choice of algorithm should not be made without competent research. Amongst the best known are DES, a symmetric algorithm that lies under a cloud of suspicion: it contains unexplained steps and there is some belief that it contains a *Back door* that would allow faster cracking by those in the know without needing a key search. AES will probably replace DES in the near future. RSA is a slower but highly trusted asymmetric algorithm. RC4 is simple, fast, and symmetric. PGP has become almost an internet standard, using a combination of symmetric and asymmetric techniques. Many encryption algorithms are subject to patents, the status of which changes as time progresses.

Summary of positive issues

A wide variety of encryption mechanisms and tools is available to implement

encryption. A proactive encryption plan will result in greater data security, customer confidence, and corporate-data integrity.

Summary of potentially negative issues

In the United States there are restrictions on the export of some encryption technologies. In 1996 the US Department of Commerce relaxed export restrictions, stating that "Any encryption commodity or software, including components, of any key length can now be exported under a license exception after a technical review to any non-government end-user in any country except for the seven state supporters of terrorism." Encryption software can not be exported from the United States to Cuba, Iran, Iraq, Libya, North Korea, Sudan or Syria. However, in 2004 several restrictions were re-implemented: "mass-market" encryption products with more than 64 bits of symmetric key require review before they may be exported and re-exported without a license. The law covers specific products and technologies and US-based companies need to understand their obligations under this legal framework. The UK and the EU have similar laws pertaining to symmetric algorithms over 64 bits and require the granting of an Open General Export License (OGEL) and prohibit export to Afghanistan, Iraq, Libya, North Korea, and China.

Regardless of legal restrictions, strong encryption algorithms are universally known, and even published in books, so a moderately competent programmer anywhere in the world could recreate them easily.

The UK also has enacted the Regulation of Investigatory Powers Act 2000, which requires that anyone who has created an encryption key must be able to produce the key if required to do so by a legal authority. This is a very controversial act, since the safety of public-key encryption systems relies upon keys not being kept by any party but the one owner.

It is easy for a programmer to overestimate the security of a new encryption scheme they have invented. Assessing the security of an encryption algorithm is an exceptionally difficult task, requiring extensive training and experience well beyond that provided by even the most advanced technical degrees. Management would be well advised not to give full rein to amateur cryptographers designing corporate security procedures.

References

- R. Needham and M. Schroeder (1978). "Using encryption for authentication in large networks of computers," *Communications of the A.C.M.*, Volume 21, No. 12.
- B. Schneier (1996). *Applied Cryptography* (New York, John Wiley and Sons).
- http://www.bxa.doc.gov/encryption/MassMarket_Keys64bitsNUp.html.

Associated terminology: Public key–private key, One-way hash, Password.

End-user development

Definition: End-user development refers to any system that is either written or configured by an end user rather than being completely developed by a systems professional for that user.

Overview

The development of computing from the 1940s until the 1980s allowed users to obtain a wide variety of processing solutions and reports. These systems were generally built and supported by a professional staff of operators, systems analysts, systems programmers, and applications programmers. This led to delays in the production of systems and in the implementation of changes, even relatively insignificant changes on a report generated by the

system. As technology became more flexible and organizational pressures demanded more flexible reporting, the delay from request to delivery became difficult to justify. In order to overcome this, technologies were deployed to allow users to develop their own applications. Some of these ran on the corporate data systems while others were stand-alone application suites that allowed users to create their own databases and applications on the corporate client–server system. For example, this allows a user at a company that has an ERP system to create their own reports to their own individual specification. End-user computing also occurs when users design (or more typically configure) applications that are then run upon a personal computer. Spreadsheets are a very common example.

Business value proposition

The ability of users to develop their own data sets and then design and run applications to provide customized reports typically reduces the demands upon the IT professionals and the IT organization overall. Usually these applications are developed as configurable modules of commercial applications. This saves the end user from having to become a technologist in addition to doing their own job, while maintaining some degree of integrity in the system being generated and used. This is important because systems frequently need to download data and communicate with other users as well as with corporate offices, so a wide mix of data types, structures, and reporting would cause more problems than end-user development solves.

Summary of positive issues

End-user development frees the corporate IT organization from having to perform each and every activity associated with user demands. There are many common applications that can be provided to users to help them develop their own solutions alongside the ability of end users to configure internal systems to meet their own specific systems needs. Applications can usually be configured to form a common basis upon which all users develop their systems, thus avoiding the problems associated with every user developing disparate systems.

Summary of potentially negative issues

It needs to be remembered that end users are usually not computer scientists or information technologists. While they may be skilled in the processes around which they develop their systems, they might not be fully aware of the issues and limitations that can have significant effects. This requires that any end-user development be done in a controlled and carefully managed environment. A further issue for organizations is that, while the technology department of the organization is freed from the obligation to perform all systems development, end users now need to be trained and have their skill sets upgraded as the technology and the organization's use of it change.

Reference

- M. Mahmood (2005). *Advanced Topics in End User Computing Series*, 2nd edn. (Hershey, PA, Idea Group Publishing).

Associated terminology: ERP, Visual Basic.

Enterprise information portal (EIP)

Definition: An enterprise information portal provides users with access to data, applications, and reporting capabilities directly from their desktop, drawing from corporate ERP, business intelligence, and data warehouse systems.

Overview

Enterprise information portals (EIPs), also known as corporate portals or *Enterprise portals*, allow corporate users direct access to the data, applications, and functionality required to perform their job function from their desktop computers.

EIPs can be incorporated as a configurable customizable desktop into an ERP environment. These EIPs provide easy access to all the users' functional requirements pertinent to performing their tasks, for example, a logistics manager providing the user with direct access to the pertinent ERP tools, business analytics, data warehouses, and communications tools required.

The role of an EIP is to provide a mechanism for searching the data in all of the organization's information systems, or as many of them as the user's clearance allows, in order to solve a problem. This has made EIPs suitable for use within the context of an ERP system, since ERP systems tie all the corporate data together through a single database. It is possible to create an EIP that works within a legacy system's environment involving disparate systems, but this is a much more difficult and technically challenging task.

Business value proposition

EIPs enable a user to have a desktop configured especially for their job function. This enables the creation of a very effective work environment that does not require users to go and search for the tools required to perform their tasks and then integrate them together. EIPs enable user-to-user communications to be tailored to the business function, reducing overhead and increasing effectiveness.

Summary of positive issues

EIPs provide users with function-specific desktops. EIPs can be built into an ERP system's configuration. EIPs facilitate access to all corporate data sources.

Summary of potentially negative issues

EIP systems are technically challenging to establish in a legacy-systems environment.

Reference

- M. Smith (2004). "Portals: toward an application framework for interoperability," *Communications of the A.C.M.*, Volume 47, No. 10.

Associated terminology: ERP, Legacy system.

Enterprise resource planning (ERP)

Foundation concept: Database.
Definition: An enterprise resource planning system is an integrated software environment that integrates processes and data using a single database.

Overview

ERP systems are designed to integrate business functions across an organization using a single shared database. The functions of an ERP are contained in a set of modules whose processes can be configured to meet organizational needs. These modules typically include accounting, finance, logistics, procurement, manufacturing, human capital management, and customer-relationship management functions. The modules in an ERP interact and communicate through a shared database, typically using a database server, in which all of the transactional data associated with the ERP is stored. The strength of the system is in part due to the nature of the database structure in that a data item is stored only once (for example, in an ERP system, a vendor's address is stored only once, whereas in a traditional system composed of multiple separate systems, a vendor's address may be stored in the database associated with the accounting system, in another database associated with the logistics system, and yet again in the customer-relationship-system database). The traditional approach was problematic because it was highly inefficient and error-prone, with data items repeated in each database potentially being inconsistent. The centralized nature of the ERP database allows the data to be accessed by any module, and, should any changes to the database be made, the

new version of the data is immediately visible and accessible by all modules.

ERP systems enable end-to-end process cycles such as the procurement-to-payment cycle and are capable of interacting with ERP systems in external organizations through XML or other file types. ERP system modules are typically run on a single operating system and the modules have a consistent user interface.

Business value proposition

ERP systems are composed of a set of modules that perform a variety of business activities ranging from logistics to customer-relationship management. The primary benefit of the ERP systems philosophy is that the system modules are designed to integrate together, interacting through the single database. The modules are configurable to allow companies to align the processes contained in the module with their own needs.

ERP systems vendors have traditionally attempted to encode best industry process practices into their systems. Consequently, ERP vendors have created industry-specific versions of their systems, e.g., healthcare, manufacturing, and retail. ERP systems allow organizations to focus upon their core business processes and decommission old, hard-to-maintain, legacy systems.

Summary of positive issues

The design of ERP systems as integrated software environments whose modules communicate through a single database relieves organizations of the burden of maintaining multiple databases and modules that may have run on disparate operating systems and architectures. ERP vendors constantly revise their systems and work to incorporate not only best practice at the process level but also the latest legal and regulatory requirements. ERP vendors also work to incorporate the latest technologies into their environments (e.g., radio-frequency identity tags), relieving the corporate IT organizations of the task.

The ERP environment allows greater security and accountability as well as better corporate governance (for example, simplifying Sarbanes–Oxley and HIPAA compliance). ERP solutions, vendors, and consultants are available to support companies of all sizes from small businesses to the largest global enterprises. ERP systems facilitate the development of data warehouses since the single database resolves many of the data-conflict issues associated with extracting data from multiple sources.

Summary of potentially negative issues

ERP system implementations can be complex and expensive, depending upon the scale of the implementation, the number of modules, the number of users, and the number of critical functions being provided. Failure to implement them correctly can result in severe consequences, so the decision to implement an ERP is a board-level decision. An ERP system must be assessed with respect to the functionality that a company will require from it. The inability to meet the critical core needs of a company might tempt organizations to customize their ERP system, something that should generally be avoided.

Several vendors have written their ERP systems in proprietary specialized programming languages and this can affect the effort required to create any specialized modules or add-on functions. Some ERP systems do not support common operating systems and databases; this can affect the selection of the systems architecture.

Reference

- D. O'Leary (2000). *Enterprise Resource Planning Systems* (Cambridge, Cambridge University Press).

Associated terminology: XML, Operating system.

Entity-relationship diagram (ERD)

Foundation concept: Database.
Definition: An entity-relationship diagram is a diagrammatic form used in the design of the logical structure of a database and as part of its documentation.

Overview

Entity-relationship diagrams (ERDs) were first developed by Peter Chen in 1976. They use a set of diagrammatic forms to represent the *Entities*, *Attributes*, and *Relationships* in a database. An entity is a class of object for which the designer (systems analyst) wishes to store data. An entity has associated with it a set of attributes that define the form that the data takes. For example, in a very simplified purchasing-department database, there might be two different kinds of data recorded: one to represent a manufacturer and one to represent a product. The attributes of the manufacturer might be name, address, contact telephone number, etc.; the attributes of a product might be the product description, its UPC, and its price per unit.

Relationships show the associations that exist between entities; thus the fact that a particular manufacturer produces a particular product is represented by a link between individual entries in the two tables, or by a relationship between the two entities.

The three main diagrammatic forms used are a rectangular box for entities, an ellipse for attributes, and a diamond for relationships. Lines between the shapes indicate how they relate to one another; the lines may be decorated in various ways to indicate particular details of a relationship (perhaps showing that each product must have at least one manufacturer associated with it).

The ERD technique involves three major phases, beginning with the *Conceptual data model*, the highest-level view of the domain, containing only the entities and the relationships that exist between them. The conceptual model is then refined into a *Logical data model* in which attributes and *Keys* (unique identifiers for individual entities) are indicated and normalization is performed. The third level is the *Physical data model*, which again refines the design to take into account physical implementation constraints. These three stages enable the systems analyst to create a design that is workable and, whenever possible, optimal.

Business value proposition

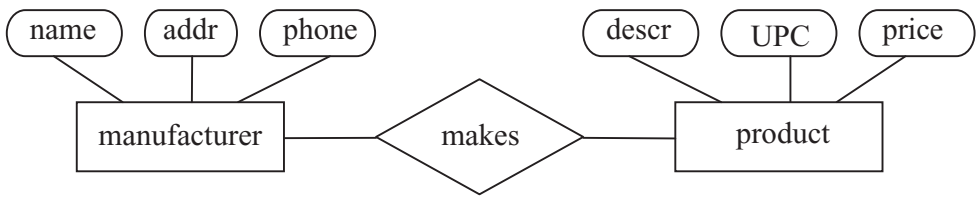
ERDs provide the systems analyst with a powerful tool through which systems can be modeled and refined, leading to the ultimate implementation. The ERD modeling technique has been used in various forms for 30 years and continues to be one of the keys to successful systems design.

Summary of positive issues

ERDs have been used by a large group of analysts and IT professionals for over 30 years. They are supported by consultants, tools, and an extensive academic and practitioner literature.

Summary of potentially negative issues

The ERD style of design is being superseded by other techniques, primarily UML, that are more amenable to the *Object-oriented* style of development and programming.



Reference

- P. Chen (1976). "The entity-relationship model – toward a unified view of data," *A.C.M. Transactions on Database Systems*, Volume 1, No. 1.

Associated terminology: UML.

Ethernet

Foundation concepts: Network, Protocol.
Definition: A common connectivity medium for local area networks.

Overview

Ethernet was designed by the Digital Equipment Corporation, Xerox, and Intel in the mid 1970s to provide a means for connecting together a number of computers in a *Local area network* (LAN), using very cheap hardware for the connections. The design was extremely successful, and is the basis of most LANs today. Although the *Network cards* required to add ethernet functionality to a computer were originally not particularly cheap, a LAN requires no other infrastructure besides extremely low-cost cables and connectors. Ethernet is now defined by the IEEE 802.3 standard.

The basic idea behind an ethernet network is that all of the computers involved are connected to a single long strand of cable, which they use in much the same way as people using "walkie-talkies"; the "ether" in ethernet refers to the (nonexistent) medium through which radio waves are imagined to travel. When one computer needs to transmit some data, it simply waits until the network is quiet – no other transmissions seem to be in progress – and transmits its own data. Inevitably this will result in occasional collisions – two computers simultaneously noticing that the network is quiet and starting their own transmissions – but this is not a problem. Every computer constantly monitors all traffic on the network; if a collision occurs, the sender will notice that there was interference, and automatically attempt retransmission after some random delay.

This kind of system is technically known as CSMA/CD: *Carrier sense multiple-access/collision detect*. It works very well with only a moderate number of computers connected to the LAN, or when overall network usage is low. As network traffic increases, the likelihood of collisions also increases, until it reaches a critical point at which attempting to transmit more data results in less data actually getting through. However, the ability to create a fully functional network without any extra support hardware is very attractive, and ethernet remains the most popular set of standards. On an ethernet LAN, each computer has a unique address (known as its *MAC* – media access control – or *Hardware address*), and all transmissions are strictly formatted to include source and destination addresses, plus error-detection codes to ensure that all collisions are detected.

The original ethernet specification used transmission speeds of 10 Mbps (million bits per second) over a single *Coaxial cable* (see *Cables and connectors* for details). Newer versions use speeds of 100 Mbps (Fast Ethernet) and 1000 Mbps (Gigabit Ethernet), and different kinds of cables; coaxial cable is not used above 10 Mbps, and is usually replaced by *Twisted-pair* cables. Coaxial cables look like the cables used to bring cable television into a home, or to connect television sets to rooftop antennæ; twisted-pair cables look like telephone cables, but the connectors are of a slightly different size. Fiber optics provides a third alternative, the cable being distinguished by the complete absence of metallic conductors.

The different versions of ethernet are distinguished by a simple naming scheme, which provides the familiar but somewhat mystical designations 10-base-2 and 100-base-T. These names always have three parts; the first is the transmission speed in

millions of bits per second; the second is always the word "base" (except in the rare case of 10-broad-36, where "broad" indicates that multiple channels of a cable-television system are used); the third indicates the type of cable used. When the third component of the name is a number, as in 10-base-2 or 10-base-5, it means that coaxial cable is used, and the number itself is the maximum length for a run of cable in hundreds of meters. So 10-base-2 means rather cheap narrow coaxial cable with no runs of more than 200 m (to be pedantic, the maximum is 185 m), and a transmission rate of 10 Mbps; 10-base-5, also known as *Thickwire Ethernet*, uses a thicker and more expensive coaxial cable to allow longer runs of up to 500 m.

When the third component of the name is a letter, it indicates a specific kind of non-coaxial cable. "T" refers to twisted-pair cable (telephone-like), and "F" refers to fiber-optic cable. A second letter or digit may be added to give more specific details: 100-base-F is Fast Ethernet using fiber-optic cables; 1000-base-T is Gigabit Ethernet using twisted-pair cables. The letter "X" is used informally to mean "anything," so 1000-base-X means all kinds of Gigabit Ethernet. "L" and "S" refer to light of *long* and *short* wavelength on fiber-optic cables.

When any kind of cable other than coaxial is used, it is no longer possible simply to connect multiple computers to the same strand of wire; additional items of hardware such as hubs and switches are needed to connect multiple systems. With twisted-pair and fiber-optic cables, each segment of cable connects only two devices, but it still works in fundamentally the same way as with coaxial cable. See *Network devices* for details.

Business value proposition

Several types of LAN technologies exist, including token ring, fiber-distributed data interface (FDDI), and ARCnet. By far the most extensive and popular network technology in use is that of ethernet, due to its low cost and the wide range of technical options for those deploying the system.

The different options available to the network designer need to be carefully considered through a set of parameters, including the speed and bandwidth requirements for the organization, the budget available, and the possible future need of the organization for an increased bandwidth capacity.

Summary of positive issues

Ethernet is an almost universal network technology. The technology is easy to implement, scale, and adapt to meet organizational needs. Ethernet technology is used and universally understood by computer systems capable of being networked.

Summary of potentially negative issues

Network systems need to be monitored and stress tests must be performed. As network traffic grows, there is a potential for bottlenecks to be created and systems performance to be reduced.

References

- C. E. Spurgeon (2000). *Ethernet: The Definitive Guide* (Sebastopol, CA, O'Reilly Press).
- W. Stevens (1994). *TCP/IP Illustrated* (New York, Addison-Wesley).

ETL (extracting, transforming, and

Foundation concepts: Database, Data warehouse, ERP.

Definition: A set of processes used to ensure that data is of a high quality, consistent, and uniform.

Overview

The effective use of a data warehouse or ERP system requires that the data in its database is of a high quality and uniform in nature. In order to achieve this, certain processes are undertaken when the data warehouse is initially created or when data

is added to it. These are the data-extraction, data-transformation, and data-loading processes of ETL.

The very nature of data warehouses, in that they are read-only data repositories intended to be used for analytic purposes and compliance archiving, requires that the data contained within them be of the highest quality. ERP systems similarly require data that is consistent and of a high quality. To ensure that these criteria are met, the data placed in the systems first has to be extracted from the source database, then examined for a variety of possible corruptions prior to incorporation, and finally placed into the database of the ERP or data warehouse.

The extraction process is usually performed by a software tool that is able to configure itself to adapt to the type (e.g., relational or hierarchical) and data structures of the originating database. The tool may need to be configured for a variety of types of legacy database that may be over 30 years old in some cases. Inside these databases there is an infinite number of possible data structures in which the data could be encoded.

The data-extraction tool must also be capable of combining the data from a set of sources, each potentially different in nature (e.g., a relational database, a spreadsheet, an indexed sequential file, and a flat file). The process of data transformation (sometimes referred to as data cleaning or "scrubbing") ensures that the data warehouse contains just one unique data element for each data item in the database. This may be necessary because the originating databases may have used different data representations, for example the warehousing database system may have used "US" to represent a country, the customer-relationship-management systems may have used "U.S.A.," and the human-resources systems may have used "United States," all meaning the same thing.

Once the data warehouse or ERP has been established, all future data that is written to the database needs to adhere to the data structures and conditions that have been established for the database, and hence must pass through a transformation process.

Business value proposition

A data warehouse that has not been populated with consistent data is useless since the whole purpose of a data warehouse is to be able to use the consolidated data to perform *Business intelligence* (also known as business analytics). Without consistent data, the old adage "garbage in, garbage out" applies and may render any analysis useless.

ETL is also used when the database for an ERP system is being created. As with data warehouses, ERP systems require a cleansed, consistent database if they are to work as designed.

Summary of positive issues

The data-warehousing and ERP industries are mature, and this has led to a wide range of vendor-supported ETL products to manage the process. Vendors range from database vendors, data-warehouse vendors, and ERP vendors to specialty third-party vendors. There are many consultants and database specialists available to support the task. Once the data has been ported over to the new database, this facilitates future system-migration efforts since the data is already clean and consistent.

Summary of potentially negative issues

The creation of a data warehouse or ERP requires the data to pass through an ETL process; failure to perform this process correctly will lead to poor and possibly catastrophic systems performance. The ETL process can be a time-consuming activity and requires significant effort to ensure that

the data is correct and amenable to future corporate needs, and that it conforms to the system requirements.

Reference

- R. Kimball and J. Caserta (2004). *The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data* (New York, John Wiley and Sons).

European Union Directive on Privacy

Definition: The European Union Directive on Privacy and Electronic Communications aims both to protect the personal data of individuals and to encourage the free movement of such data by having EU member countries ensure the rights and freedoms of the individual to data privacy during data processing.

Overview

The European Union Directive on Privacy and Electronic Commerce, which contains 21 articles, was adopted on July 12, 2002 and an implementation requirement was adopted by member states on October 31, 2003. The directive supplements the Telecommunications Data Protection Directive (97/66/EC), also known as the Communications Data Protection Directive (CDPD).

According to article 1, "The directive harmonizes the provisions of the Member States required to ensure an equivalent level of protection of fundamental rights and freedoms, and in particular the right to privacy, with respect to the processing of personal information in the electronic communication sector and to ensure the free movement of such data and electronic communication equipment and services in the country."

Key articles include the following.

- Article 2 covers "the processing of personal data in connection with provision of publicly available electronic communications services in public communications networks in the community," and digital exchanges and subscriber lines connected to analog exchanges.
- Article 4 covers the security requirements and calls upon providers not only to take the appropriate "technical and organizational" security measures, but also to notify their subscribers of the risks.
- Article 5 calls upon member states to "ensure the confidentiality of communications and related data traffic."
- Article 7 defines the rights of subscribers to have itemized billing for services.
- Article 8 expresses the obligations of service providers regarding the presentation of calling-line identification.
- Article 9 discusses the issues surrounding location data.
- Article 12 defines the rights of subscribers in relation to the capture of subscriber data in public directories.
- Article 13 covers unsolicited communications such as direct marketing via email. The article states that email in which the sender disguises or conceals their identification is prohibited.
- Article 14 states that there are no mandatory equipment requirements but that equipment should be compatible with fulfilling the other articles of the directive.

Business value proposition

The European Union directive is a powerful and wide reaching initiative. The directive has a different perspective from the US legislation. In the EU the principal philosophy is that an individual's rights come first and that they must *opt in* to any data set rather than be automatically included in a

data set and than have to *opt out* if they desire. This prevents the merger of data sets within the business community without the prior consent of the individual.

The 2002 directive built on an earlier directive (95/46/EC) that discusses the controversial issue of inter-country data transfer. Article 25 of 95/46/EC states that "The Member States shall provide that the transfer to a third country of personal data which are undergoing processing or are intended for processing after transfer may take place only if, without prejudice to compliance with the national provisions adopted pursuant to the other provisions of this Directive, the third country in question ensures an adequate level of protection." So far the EU Commission has recognized five countries as meeting the provisions of the article: Switzerland, Canada, Argentina, Guernsey, and the Isle of Man, plus the US Department of Commerce (DOC) through its "Safe Harbor Privacy Principles," and the US Bureau of Customs and Border Protection for the transfer of air-passenger name records in order to provide adequate border protection.

The United States has proved a difficult market for the EU to ignore with respect to data traffic, since US corporations operate in a "sectoral" framework that "relies on a mix of legislation, regulation, and self regulation" (US DOC) and are clearly not subject to EU law. A "safe-harbor" compromise was developed, in which organizations can either join a self-regulatory privacy program that adheres to the safe harbor's requirements or develop their own self-regulatory privacy policy that conforms to the safe harbor (US DOC).

The safe-harbor policy includes components pertaining to the following issues.

- *Notice*, notifying and informing individuals about data collected and providing recourse for individuals who wish to remove or modify their own data.
- *Choice*; individuals must be given the opportunity to choose (opt out of) whether their personal information will be disclosed to a third party or used for a purpose incompatible with the purpose for which it was originally collected or subsequently authorized by the individual. For sensitive information, an explicit affirmative choice (opt in) must be given if the information is to be disclosed to a third party or used for a purpose other than its original purpose or the purpose authorized subsequently by the individual.
- *Onward transfer* pertains to the disclosure of information to a third party, stating that organizations must apply the notice and choice principles.
- *Access* covers the rights that individuals have pertaining to their ability to correct, amend, or delete information when it is inaccurate.
- *Security* pertains to the "reasonable precautions" safe harbors must take to protect personal information from loss, misuse, and unauthorized access, disclosure, alteration, and destruction.
- *Data integrity* pertains to the relevancy of the use of data in terms of the reason for which it was collected; and section viii details enforcement (US DOC).

References

- Official Journal of the European Communities, L 201/37, 2002.
- US Department of Commerce (2000). *Safe Harbor Privacy Principles, Issues* (Washington, DC, US Department of Commerce).

Associated terminology: Law cross-reference.

Fiber optics

Foundation concept: Cables and connectors.
Definition: Data encoded as pulses of light conducted along flexible transparent wires as a replacement for data sent as electrical pulses along a flexible metal wire.

Overview

The most familiar means for transmitting signals or data is by sending electrical impulses along a flexible metal (nearly always copper) wire. This has many advantages: electrical signals are the native internal infrastructure of all computing devices, so very little signal conversion is required; wires can easily be insulated to prevent cross-over with other signals; electronic signals travel along wire at the speed of light, so speed is maximized. There is also one major disadvantage: when electric currents pass through metal wires they generate magnetic fields, and when metal wires pass through varying magnetic fields, they pick up induced electric currents. This means that, whenever two signal wires run close together, they will interfere with each other, and whenever a wire passes near to any electronic device, it will pick up interference from it. Heavy-duty electro-mechanical devices, such as large electric motors, can easily generate enough interference to swamp any real signal, and even enough to damage connected electronic equipment.

Common alternatives, such as transmitting signals by radio or microwave, suffer from similar problems. Radio and microwave signals can not be insulated and can not be channeled along a "wire," so multiple signals have to use significantly different wavelengths in order to avoid complete interference; electrically generated interference is still a problem for radio, and microwave signals need a straight "line-of-sight" unobstructed path to follow; and there may be significant environmental dangers associated with some wireless signals (there is some controversy, but microwave radiation in the environment is a major concern for many). Converting between electronic signals and radio or microwave signals for transmission requires expensive equipment.

Fiber optics provides another alternative. Instead of an electrically conductive wire, a light-conductive (i.e., transparent) fiber is used. A light shone down one end will follow curves in certain kinds of transparent fibers, and appear at the other end. It seems as though the light follows a curved path, although it is in fact following a large number of short straight-line paths, gradually being reflected around curves. For long-distance transmissions, the fibers are very narrow strands of glass, and the light used is infrared (invisible to the human eye) laser light. Laser-light emitters can be designed to switch on and off at exceptionally rapid rates.

Fiber optics are successfully used to transmit signals at almost 100 000 000 000 bits per second, a far greater bandwidth than is available with metal wires, over distances of up to 100 miles, a distance that would require a number of retransmissions to maintain an electrical signal. Fiber optics do not interact with magnetic fields, and do not create them to any detectable degree, so a fiber-optic cable could run through the center of the largest motor in the world without picking up interference, and thousands of fibers can run together for tens of miles without any signal cross-over. Fiber-optic cables are much lighter than metal wires, and it is much more difficult to tap into them and intercept signals. Fiber optics are also used for short-distance communications, even between components of the same computer system, because of their high bandwidth capacity and immunity to electrical noise. *FDDI* (fiber-distributed data interface) is a well-known implementation for LANs (local-area networks) that use fiber optics instead of metal wires, and allows systems on the same LAN to be over 100 miles apart.

The disadvantages of fiber optics are that the cables are more expensive, and the equipment needed at either end of a cable to convert between light and electronic signals is much more expensive. Fiber optics can only carry signals; optical computing devices, although of great theoretical interest, do not yet exist. Connecting together two fiber-optic cables is very difficult. It is also impossible to transmit any useful amount of power along a fiber-optic connection, so all connected devices must have their own power sources.

Business value proposition

Fiber-optic cables are advantageous for several compelling reasons. The first is their ability to carry a far greater bandwidth than is possible over any other medium. Second, they do not suffer from any electromagnetic interference (EMI) and hence can be incorporated into system designs where metal cables would either be impractical or require special and expensive treatment (e.g., automotive and aeronautic engine-management systems). Third, the cables permit data to be transmitted over large distances without the need for signal boosters. Fourth, the cables themselves are much lighter than metal wire, and thus are favored in weight-sensitive applications (e.g., in airliners).

Although the price of the cable is higher than that for metal-based cables, the versatility and advantages of the technology will increase its use and, as demand rises, costs of supply will drop. Eventually fiber optics will probably become the de facto standard media for data transmission.

Summary of positive issues

Fiber-optic cables are light-weight, have a high-capacity bandwidth, are not subject to EMI, and can carry signals for up to 100 miles without the need for retransmission or a signal booster.

Summary of potentially negative issues

Fiber-optic cable technology (the cable itself and the light–electronic signal converters) is more expensive than traditional metal cable technologies. Connecting two cables together is difficult. All devices connected to the cable need their own power supplies because no useful power may be transmitted through a fiber-optic cable.

Reference

- D. Goff (2002). *Fiber Optic Reference Guide*, 3rd edn. (Woburn, MA, Focal Press).

Associated terminology: LAN, Ethernet.

File server

Foundation concepts: Network, Client–server.
Definition: A computer or group of computers set aside to act as a data repository, storing files for many users, and making them accessible over a network.

Overview

When it was normal for organizations to have one main computer, which was used by everyone who needed computer resources, all files were kept on the same system, and were, in principle at least, accessible to all. Now that everybody has their own computer, sharing files and accessing one's own files from a different location has become an important concern. Portable removable media (such as flash memory cards) are a partial solution, but require that the need for a particular file is anticipated. The various computers on a network could all be set up to run FTP (file transfer protocol) servers so that they can be retrieved by the network from anywhere, but that adds to the administrative load, and creates a significant security risk.

The other solution is to provide a *File server*. This is one or more specially designated computers on a network that are responsible for storing files securely and

making them available to authorized users. Normally, when a file server is in use, the users' own computers cease to be the primary storage site for their own files; everything of importance resides on the file server, it is downloaded when work starts, worked on locally, then uploaded back to the file server when finished. Many users prefer to keep their own files on their own computers, perhaps wisely not trusting all their eggs to one basket; they must simply remember to upload up-to-date copies of their files to the server at reasonable intervals.

The use of a file server is a great aid to system maintenance: creating regular backups of all files is much easier if all the files are in one place. File servers *can* also provide financial gains: if all large files are kept on a file server, then the individual workstation computers can have much smaller and therefore cheaper disks than would normally be possible. Some organizations go all the way to using *Diskless* systems, in which the individual workstations have no disks at all, relying on network access to a file server for absolutely everything.

There are some negative aspects to the use of file servers. If all files reside on a network-accessible file server, then the security of the entire network and of that server in particular becomes a major concern. Network traffic will be greatly increased if every file access requires communication with a remote file server, so higher-bandwidth network infrastructure may be required. Even the fastest of networks can not provide the same access speed as that available from a local disk drive, so there may be an appreciable slowdown in some applications.

The term file server may be correctly used to refer either to the computer that holds the files, or to the software application that runs on that computer handling the requests. File server software is technologically fairly simple, so it is not difficult to find reliable implementations from a variety of sources. Using a file server can also be an equally simple procedure. In popular systems, a *Shared folder* is used. This appears as a perfectly normal folder on the desktop, which can be opened and accessed just like any other folder, but in fact represents an automated portal to a file server. For example, adding a file to the shared folder is translated behind the scenes into the correct sequence of network commands to send that file to the server. Many users happily use a network file server for years without ever realizing it. *NFS* (*Network File System*, not to be confused with *NSF*, the *National Science Foundation*), originally created by Sun in 1984, is one of the oldest and most popular of current systems. Versions of it are often freely available for most versions of Unix and many other platforms.

Business value proposition

File servers are used in organizations for a variety of reasons, including centralized security, user management, application license management, cost, backup processes, efficiency, and integration. Having all the files on one server allows that server to be at the center of all security efforts; all requests for data can be examined equally and verified, preventing localized security breaches and duplication of effort in securing many localized data storage facilities. An extension of centralized security is the centralized control of user access and the creation of user profiles (descriptions of what resources each user is permitted to access). This can help in the creation and enforcement of access policies and ensure compliance with regulatory requirements such as Sarbanes–Oxley.

The use of file servers allows network managers to ensure that the applications running on corporate systems are in accordance with the licenses held by the organization. The centralization of licenses also

allows managers to ensure that the licenses are being used optimally and effectively. This is important because many application vendors charge customers on the basis of the number of users accessing the application or the number of processors used.

File systems allow centralized backup systems to be efficient and effective, ensuring that the correct processes are performed. The use of file servers allows network managers, CIOs, and chief security officers to manage the complete information-management lifecycle.

File servers allow the creation of either thin- or fat-client systems. This allows an organization to design an efficient and integrated system that reflects its own individual needs.

Summary of positive issues

File servers provide centralized data storage and access capability. File servers allow systems architects to develop a wide range of system configurations and help network mangers control costs, security levels, users, licenses, backups, and the efficiency of the system. Many freeware and open-source file servers are available and provide a variety of functions.

Summary of potentially negative issues

File servers can be complex to manage and require special training to ensure correct use. The problematic nature of "putting all your eggs in one basket" is a critical concern. If all of a corporation's data is stored in one place, then that one place needs to be very carefully protected.

Reference

- M. Tulloch (2003). *Windows Server 2003 in a Nutshell* (Sebastopol, CA, O'Reilly Press).

Associated terminology: FTP (file-transfer protocol), Client–server, Network.

File system

Foundation concept: Disk.

Definition: The logical organization of portions of a disk or other long-term storage device needed to create data files.

Overview

A standard disk drive contains hundreds of thousands of millions of bytes of data, enough to store 100 000 copies of the Bible without compression. The only inherent organization is that those bytes are grouped into hundreds of millions of *Blocks*, each of exactly 512 bytes (enough for just one small paragraph of text). If users are to be able to store a large number of data files of various sizes, large and small, and be able to find them again, some additional organization must be imposed upon a disk. That is exactly what a file system does.

One of the major components of an operating system (such as Unix, Windows, DOS, etc.) is the built-in software that implements the file system and gives users and applications convenient and reliable access to it. An operating system can not really exist without a file system, since an operating system must reside on disk as a large collection of essential files. In general, each different operating system has its own kind of file system, and a disk that was formatted for one system may be completely unreadable by another, although many systems do now provide some cross-system support.

All file systems provide the same basic functionality: creating and accessing files and folders (outside of Windows and MacOs, folders are usually called *Directories* or *Catalogs*). However, different file systems do vary considerably in their provision of the more advanced features which may be considered essential by any business venture or serious user, such as control of access by multiple users, and recovery after disk "crashes." The major file systems commonly in use are listed below.

FAT (File Allocation Table), which has three varieties: FAT-12 (used only for floppy disks), FAT-16, and FAT-32. The number 12, 16, or 32 in the name refers to the number of bits used in the unique identification number of each file. For example, FAT-12 uses only 12 bits, which can produce only 2^{12} or 4096 different numbers, so, under the FAT-12 system (and therefore on any PC floppy disk), it is not possible to have more than 4096 files. The same logic imposes a limit of 65 536 files on any FAT-16 system. This restriction, together with the maximum of 11 characters in a file name and 4 GB maximum disk size, has rendered FAT-16 obsolete. FAT-32, introduced in 1996, relieves some of these limits, supporting disks of up to 2048 GB with individual files of up to 4 GB and the now-standard long file names. However, FAT systems provide hardly any support for recovery after disk crashes, and even the unexpected loss of electrical power can result in enormous loss of data. FAT systems also suffer from *Clustering*, a technique that significantly raises the maximum supportable disk size at the expense of also raising the minimum possible file size. A normal FAT-32 system on a large disk will require every file, even if it contains only a single byte of data, to occupy at least 32 768 bytes of disk space. FAT-32 is generally available only under Windows.

NTFS (the Windows-NT File System) is a Microsoft product introduced for Windows-NT, and supported by all modern Windows variants. It provides faster access to data in large files, automatic encryption, logging of file accesses at various levels, individualized file-access protection settings, and good recovery after disk, computer, or software failures. It is the file system of choice for Windows-XP.

Unix. All versions of Unix (Linux, FreeBSD, Solaris, etc.) use minor variations on the same file-system design which, surprisingly, has no name (apart from **UFS**, which simply stands for Unix File System). Its most recognizable and namable feature is the *I-Node*, a small data structure used to represent an individual file. I-Nodes are probably also the most troublesome feature, since they must be created in advance when a disk is formatted; if the number of I-Nodes needed was underestimated and they are all used up, it is impossible to create any new files. Unix file systems generally allow for very long file names; and always provide individual access permissions on files and directories.

ISO9660 is the file system used for data on compact disks (CDs). It has two forms: Level-1, which allows only 11-character file names; and Level-2, which allows up to 32-character file names. Both permit folders to be nested only eight deep. **Joliet** is a commonly used set of extensions that greatly relieves these restrictions. **Micro-UDF** is similar in spirit to ISO9660, and is used for data DVDs. **UDF** (the Universal Disk Format) is another newer extension to ISO9660, which is used only for optical disks (CDs, DVDs, etc.)

HFS (Hierarchical File System) and **HFS-Plus** were the file systems used by Apple Macintosh computers before MacOS-X, which is based on Unix (the BSD version). They support long file names, file access permission setting, and the more complex file structures that Macintosh systems were famous for, but have other limitations similar to those of FAT systems.

Business value proposition

The systems administrator must understand the underlying structural limits, strengths, and weaknesses of the file system in use as part of an organization's systems architecture. A file system is an integral part of an operating system, and the whole is a very sophisticated piece of software that continues to evolve; it is essential for administrators to keep abreast of the latest technology applicable to their systems.

Operating systems are increasingly designed to provide robust file systems that can survive hardware failures and operator errors, and can even be configured to provide warnings of impending problems (such as running out of storage space, or problems with the disk hardware). Unfortunately, system problems and instabilities tend to occur more frequently at the edges of a system's performance limits, and, when a system is operating under extreme stress, failure can occur without any detectable precursors. It is an operator's responsibility to manage the system proactively.

References

- A. Silberschatz, P. Galvin, and G. Gagne (2004). *Operating System Concepts* (New York, John Wiley and Sons).
- B. Carrier (2005). *File System Forensic Analysis* (Boston, MA, Pearson).

Associated terminology: Operating system, File server.

Firewall

Foundation concepts: Network, Virus, Spyware, Internet protocol.

Definition: A device that confidentially monitors all applications attempting to establish internet communications, and stops those that meet predefined standards of suspicious behavior or that fail to satisfy predefined standards of acceptability.

Overview

A *Firewall* may be a separate hardware unit installed directly between a computer and its internet connection, it may be an integral part of another item of network hardware such as a switch or broadband modem, or it may simply be a piece of software that works with the operating system that has already been installed.

Whatever form it takes, the job of a firewall is to monitor and police network activity. Not spying on the content of transmissions, and never recording them, but rather making judgments based on the source, destination, and protocol of individual messages.

If a particular computer is threatened by frequent break-in attempts, and it does not provide any public network services, a simple solution is to instruct the firewall to block all incoming connection attempts. Communications initiated from the inside will be unaffected, but those initiated from outside (as *most* break-in attempts are) will never be seen.

If a threatened system does provide some public network services, perhaps acting as a web server or email server, the firewall can be instructed to block all ports except those directly associated with web and email services (i.e., ports 25, 80, 109, 110, 143, 993, and 995) from receiving connection attempts. Of course, it is then the system administrator's duty to ensure that the applications responding to those ports are secure and trustworthy.

Firewalls can also be used to prevent outgoing connections, so, for instance, children or employees could be prevented from browsing the web, accessing off-site email servers, or playing with instant-messaging systems. If a system has been infected by a virus, it is possible that break-in-attempting connections can be initiated from inside the system; some firewall products can be programmed to report unusual connections, and ask the user "forbid or permit?"

on a case-by-case basis for anything unexpected.

The primary purpose of firewalls is, of course, to restrict communications into or out of a controlled area. If used carefully, they can prevent attacks from getting in and secrets, corporate or personal, from getting out. Firewalls can be a heavy-handed solution; employees often bemoan the fact that they can not use FTP and therefore can't download the latest version of some application, but that is just the firewall doing its job especially well. It is more often firewall-management policy that is at fault rather than the firewalls themselves.

Of course, a firewall is just another human construct, as fallible as any other. If a firewall system itself has security vulnerabilities through which it can be taken over, then all is lost. Single-purpose firewall hardware units are the least likely to have such vulnerabilities, but are by no means to be considered immune. Software firewalls are certainly the most vulnerable, since they are merely software applications themselves, and are just as likely to be replaced or modified by a successfully attacking virus. For any system holding critical data, the selection of a firewall is a critical stage in setting up security, and requires diligent research: some providers have expensive products and a long history of satisfied customers; some providers are very cheap, but have never been heard of before. A small saving at setup may have a very large cost later. A vulnerable firewall means an open computer system.

Business value proposition

The key to a secure IT operation is to ensure that unwelcome traffic stays outside the walls of the organization. If there is any possible doubt about the security of a computer system (and with closed-source systems how can there not be a doubt?), then the provision of a reliable firewall system is central to safe operation. Firewalls examine network traffic and resource requests, assess the threat level, and take the appropriate action. The application of firewall technology is the responsibility of the network manager and the chief security officer (CSO) (in a small organization the CSO may well be the network manager). An appropriate firewall technology and vendor need to be selected and put in place. In the selection of a firewall, the network manager needs amongst other things to consider the reliability of the device, its costs relative to performance, whether a hardware or software device is appropriate, the scalability of the selected system, vendor or third-party support for the device, the historical record of the vendor, and the total cost of ownership.

The implementation of a firewall device needs to take into account the sensitivity requirements of the network to which it is attached and the requirements of its users. Should all traffic be prevented from entering the system or just select traffic? Insensitivity will allow too much traffic and the potential for breaches in security; if the sensitivity levels are too high users will be unhappy because "normal" emails may be rejected. These aspects of deployment need to be contained within a firewall-management policy document.

Summary of positive issues

Firewall technology helps to prevent unwanted data traffic from entering a network. The technology is mature, well understood, and supported by vendors, consultants, and network specialists. Firewalls can be used to prevent access to specific resources.

Summary of potentially negative issues

Firewalls are not immune from attack themselves, and software firewalls in many instances may themselves be targets of hostile intent. Poor firewall-management policies can result in restricted access to the

network both for internal and for external users.

Reference

- J. Wack, K. Cutler and J. Pole (2002). *Guidelines on Firewalls and Firewall Policy: Recommendations of the National Institute of Standards and Technology*, US Department of Commerce Special Publication 800–41 (Gaithersburg, MD, US Department of Commerce).

Associated terminology: Proxy, Security.

Flash memory

Foundation concepts: Storage, Memory, Disk.
Definition: A form of stable memory maintaining its contents indefinitely without any power consumption, and having the robustness of a solid-state device, with no moving parts.

Overview

In an ideal world, the memory used in a computer would have five essential characteristics: it would be

(1) of low cost,
(2) of high capacity,
(3) of high speed,
(4) non-volatile (meaning that it should not forget its contents when the power supply is turned off), and
(5) robust (both solid state, having no moving parts that will inevitably suffer from wear and fail, and having no delicate components).

Naturally, the real world is far from perfect, and no form of memory yet devised satisfies all five of the requirements. Traditionally, the problem has been solved by building computers with two forms of memory, one (usually called RAM) that satisfies requirements (3) and (5), and the other (usually in the form of a disk drive) that satisfies requirements (1), (2), and (4).

Flash memory is a form of *EEPROM* (electronically erasable programmable read-only memory) that satisfies requirements (4) and (5). It is a relatively new technology, and advances toward the first three requirements are being made, but there is still a long way to go before flash memory can replace either RAM or disks in general-purpose computers. At the time of writing the best statistics for commercially available units are very approximately as follows:

Criterion	Units	RAM	Flash	Disk
Cost	per GB	$100	$100	$\frac{1}{2}$¢
Capacity	GB	2	4	500
Speed	ns per byte	0.5	50	250
Non-volatile	–	No	Yes	Yes
Robust	–	Yes	Yes	No

(In this table, each technology is given its best possible circumstances. For example, flash-memory cards can attain transfer rates of 20 MB per second, which gives 50 ns per byte, even though a single byte can not be accessed in 50 ns; similarly, disk drives can provide one byte in an average of 250 ns only as part of a large transfer.)

Currently, flash memory provides a useful portable medium. Data may be transferred to it very quickly and conveniently, and it is very light and hard to break, so it may be transported with great ease. Usually, flash memory is in the form of small cards or other portable devices. The cards must be connected to a computer through a special USB device; the popular (and mutually incompatible) forms are *Compact flash* (two different forms), *Secure digital*, and *Memory stick* (two forms). Other flash memory devices have a built-in USB plug, so they need no extra support.

There is one further disadvantage to current flash memory technologies. Although they are completely solid state, the procedure required to erase and write data is slightly destructive, and flash memory devices can not be expected to survive more than one million cycles (fewer for many kinds). Reading data has no destructive

effect, only erasing and writing. This limitation is of no significance to current uses, but, if flash memory is ever to be used to replace RAM, for which a million read–write cycles occur in less than a second, this will be another hurdle to be overcome.

Business value proposition

Flash memory has many desirable properties for a memory device. It is capable of holding a meaningful amount of data, as opposed to the floppy disk which has become almost redundant due to its small storage capability, which forces users to employ compression or other techniques to save even modest files. Flash memory has relatively fast data transfer rates, and most useful of all is that it is portable and can employ the convenience of a USB connection. These characteristics make it ideal for the business traveler or individual who wishes to carry their data with them on a trip.

Security concerns also need to be addressed by flash memory users; the devices have no inherent security built into them and thus users need to ensure that the data stored is encrypted as necessary (e.g., to meet HIPAA requirements). Of course, the same security concerns apply equally to other portable media. The small size of the memory devices aids their portability and can enhance the data security because users can wear the device on a cord around their necks and thus reduce the potential for data theft; a laptop is a much more obvious target for a thief than is a hidden flash memory device.

Summary of positive issues

Flash memory is convenient, portable, easy to use, possesses a relatively high transmission rate, can hold more data than a floppy-disk device, and is solid state in nature.

Summary of potentially negative issues

The cost of flash memory is higher, per unit of capacity, than that of disk memory, and flash memory stores considerably less data. There is no built-in security associated with the devices.

Reference

- P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni (1999). *Flash Memories* (Boston, MA, Kluwer Academic).

Formal methods

Foundation concept: Software development lifecycle.
Definition: Mathematically rigorous software development methodologies.

Overview

Software development is sometimes viewed as a science, sometimes as an art, but mostly is considered to lie in an ill-defined variable position somewhere between the two. The scientific method is based on a continuous cycle of observation, theorization, and experimentation. Observation corresponds to investigating what is needed in new software, or observing the results of tests of existing software. Theorization corresponds to working out what those observations mean, and thinking of a way to implement or improve the implementation of a solution. Experimentation corresponds to running or testing the implementation, and seeing what happens in that test brings the cycle back to observation.

The artistic method is based on experience and the intuitive expertise that comes with it: the experience of having implemented many software projects before, and knowing which techniques work well under which circumstances, and the academic experience of having learned the "right" way to do things and knowing how to apply that knowledge. Neither method can be successful alone; successful software developers slowly discover their preferred place on the art–science spectrum.

Neither method is particularly suited to getting programs right. The artistic method is used because we don't know any better; programming hasn't really been "worked out," and never can be. There is no "correct" method for producing a program to solve a problem, so humanistic techniques relying on flashes of inspiration, experience, and even superstition are inevitable. The scientific method sounds ideal, but it is designed for real scientific research, investigating the real natural world, where absolute knowledge is not possible, and the best we can do is invent theories that explain all experimental results, and keep on testing those theories with new experiments.

Programming is not part of the natural world. It is a totally artificial activity, and absolute knowledge *is* possible. People invented computers; we know exactly what they do under all circumstances. People invented programming languages; we know exactly what effect every correct program will have under all circumstances. People write programs; so we can work out exactly what they will do under all circumstances. There is nowhere where experimentation can reveal anything that we could not already have been certain of; the scientific method is totally inappropriate.

Mathematics is the language of artificial, abstract, completely known systems. Why is two plus two equal to four? Simply because the laws of mathematics make it so. If one had a box with two apples in it, then added two more apples to the box, one would strongly expect to have a box with four apples in it. That is not why two plus two is four; in reality it is not much more than a coincidence. We have a very strong belief that the laws of mathematics do model the way the world works, but mathematics is in no way dependent upon that belief. If it turned out that our box actually had five apples in it, that would not mean that two plus two is not four, or that mathematics is wrong; it would only mean that mathematics might not be as practically useful as we had imagined.

Modern mathematics is the result of thousands of years of careful study, and is a tailor-made methodology for investigating, developing, and manipulating knowledge in artificial, abstract, and completely known systems. In other words, mathematics is the ideal system for software development. *Formal methods* are techniques that allow all stages of software development to be represented in purely mathematical terms, which can be manipulated to produce a program in ways that are guaranteed to be correct.

The first stage in formal software development is to produce a *Formal specification* of the problem to be solved. This means that a mathematical model for the problem domain is constructed, and logical rules describing inputs to a problem and exactly how a correct solution is related to them are defined. All further work is based on this model and these rules; they provide the definition of correctness for the program to be produced. This is a very difficult stage, requiring fairly intensive training in abstract algebra and mathematical logic. It is made more difficult by the fact that the customer must agree that the formal specification does indeed describe the problem that they want solved, and it is most unlikely that the customer will be able to understand the specification at all.

The next, and final, stage is to transform the formal specification into a program. The formal specification is by definition correct; the only thing that stops it from being a solution in itself is the fact that it will not naturally be in the form of an efficiently computable algorithm. For example, part of the specification might demand "the smallest positive number that satisfies [some condition]"; that would be a correct description of the result, but does not say how to find that result. It must be transformed into an explicit and executable form such as "take each number 0,

1, 2, and so on without limit, testing each in turn to see whether it satisfies [some condition]; when one is found, stop the search because you've got the answer." This describes the same result, but this time also says how to find it.

The well-understood method of mathematical proof supports exactly that kind of transformation. It repeatedly applies small transformations that are known to preserve correctness to an initial statement. If the initial statement was correct, the result will also be correct. The formal specification is transformed into an executable program by applying a carefully chosen sequence of simple changes that have already been proven to preserve correctness. This is also a very difficult stage. There are software development tools that will apply the chosen transformations to the guaranteed correct specifications, removing the drudgery and the possibility of transcription errors, but at each step there will be an enormous number of possible transformations that could be applied, and it is absolutely impossible for a software tool to know which is the "right" one.

Even formal methods rely completely on human intelligence. The difference is that any program produced by formal methods is absolutely guaranteed to be correct. What is not guaranteed is that any effectively executable program will be produced at all. Heuristic searches and other techniques of artificial intelligence can help to automate the search for the right transformations, but it is logically impossible for there to be any method that will always find the answer. Even human programmers can't guarantee to be able to program a solution to every problem they might be given.

The potential gains from making formal methods easier to use are enormous. The ability to produce guaranteed perfectly correct programs just sometimes would be worth a lot. As a result there has been a lot of research into formal methods, and some variably useful tools have been produced to aid the process. The most promising of these are Z (pronounced "zed"), VDM (the Vienna Development Method), The Hoare Logic, the ACL2 Theorem Prover, the HOL Theorem Prover, CSP (Communicating Sequential Processes), and Extended ML, although there are many more. Some non-traditional programming language paradigms, particularly pure functional programming and logic programming, transform specifications that would normally seem to be completely abstract and unexecutable into directly executable programs.

Business value proposition

The need to create programs quickly and correctly is at the very center of computing. The promise of perfect programs, flawlessly working and optimized, first time and every time, is the ideal sought by programmers, IT-project managers, and everyone connected to an IT organization, including the end users. However, this utopia is rarely, if ever, reached by anyone, even those organizations certified as Level 5 on the Software Capability Maturity Model.

Typically software is developed in a very ad hoc manner. For example, some code may be taken from a previous project, edited, tested, and then added to. The processes may follow a rough informal process model, e.g., a waterfall model, or a spiral model, or be termed a "prototyping" process model and couched in JAD terminology. Fundamentally, many professional and trained computer scientists acknowledge that, if houses, bridges, and high-rise buildings were constructed to the same standards as the majority of software, they would either fall down during construction or be condemned as unsafe and unfit for occupation.

The rise of formal methods as a discipline grew from the recognition that the profession needs to move away from the

position where anyone can claim to be a programmer and proceed to sell bad code to the public, toward becoming an engineering profession. Formal methods aim to provide the professional programmer with a set of tools and methodologies that, when used correctly, will result in systems that match their specifications, as agreed upon by the software engineer, the user, and, when appropriate, the corporate management.

The use of formal methods in IT organizations is very limited, and there are several reasons for this. The first is that the training to become a true software engineer is long, technically challenging, and requires intellectual rigor. Second, potential software engineers are put off from studying this topic because the demand from industry is low and the rewards consequently limited. Third, companies do not recruit software engineers because many IT organizations do not understand what true software engineering offers them. A mistaken view is that adhering to a process model such as MIL-STD-2167A, using the waterfall method, or using RAD methods equates to software engineering. This is simply wrong, weak development methods that do not encapsulate mathematical design principles are not formal methods of software engineering. Fourth, IT organizations are typically not willing to change their development model to this higher standard, since it demands a higher caliber of human resources, is perceived as taking longer to develop code, and is unintelligible to non-technically trained developers, vendors, and users.

Some organizations have attempted to use formal methods, with varying degrees of success. Formal methods are particularly suitable to problems for which failure may threaten human life, e.g., in the space program, the nuclear industry, and airlines. However, until the computer industry as a whole demands that its workers possess true professional qualifications in the way that professional engineers, chartered surveyors, and others do, with a requirement for continuous training and acknowledgement of liability for errors, the current ad hoc situation of random development will continue.

Summary of positive issues

Formal methods are the subject of much current research and great academic interest; successful use of formal methods would provide great benefits to an organization. The Software Capability Maturity Model is available to measure an organization's progress from the informal to the repeatable formal style of development. A wide range of formal methods exists, including functional programming and proof systems. Formal methods enable the user and the software engineer to agree on a specification of the product that will be delivered. Formal methods permit proof that the specification and the delivered code are the same.

Summary of potentially negative issues

Formal methods are very much an evolving technology. There is a large academic literature base, but few usable software development tools that truly adhere to formal methods. Formal methods require intense specialist training. Formal specifications are difficult for users and customers to read and hence be able to comment upon or give informed assent to. Adoption of formal methods requires different development processes from those of non-formal styles of development.

References

- W. Gibbs (1994). "Software's chronic crisis," *Scientific American*, September.
- A. Harry (1996). *Formal Methods Fact File, VDM and Z* (New York, John Wiley and Sons).
- C. Jones (1990). *Systematic Software Development Using VDM* (Englewood Cliffs, NJ, Prentice-Hall).

Fortran

Foundation concepts: Programming language, Compiler.
Definition: A primitive programming language.

Overview

Fortran was one of the first programming languages ever created, and is easily the record holder for longevity, being still in use today. The original design and implementation was begun by John Backus of IBM in 1954, but it was many years before the language was usable, and years more before it became available to the computer-using public. Compilers were unknown at the time, so the implementation of Fortran required the invention of a whole new software technology, and gave birth to a new area of research. The name "Fortran" is a contraction of "Formula Translator"; in the 1950s programming was mostly a matter of translating mathematical formulæ into a form that could be directly executed by computer, so the name is apt.

To call Fortran a very primitive programming language is a true statement, but not a fair criticism. When it was designed, it was far in advance of anything else. Programming was in its infancy, and the techniques that are now well known as the basis of successful programming had not yet been thought of. Fortran would be eminently unsuitable for any new software development – it does not even support structured programming – but it served its purpose admirably.

There is still a considerable amount of legacy code, programs that were written long ago in Fortran, and for one reason or another never got recoded in a newer language. For a long time, Fortran was the most efficient programming language for scientific calculations. It incorporated many optimizations that were not considered suitable for a general-purpose programming language, but good implementations of modern compiled languages will include these optimizations and more. Traditional Fortran is also very easy for an experienced programmer to learn, so there is little risk of this stockpile of legacy code becoming unusable. Fortran is still used by older scientists who learned to program in Fortran and nothing else, and are unwilling to learn a newer language when their old favorite still suits their purposes as well as ever.

Over the years, Fortran has evolved through a number of standards. The one known as *Fortran IV*, defined in 1961, is true to the original vision of the language, whilst fixing all of the "teething troubles" of the language, and is generally considered to be the *Real Fortran*. *WatFor* was a famous implementation of Fortran IV from Waterloo University in Ontario, which was very popular because it gave comprehensive and meaningful diagnostic error messages. *Fortran 66*, *Fortran 77*, and *Fortran 90* were later major revisions (each named after the year of their design), the last two especially trying to bring Fortran into the modern world by introducing the possibility of structured program design and object orientation. However, these changes went against the fundamental nature of the language, and many programmers took the reasonable attitude that it would be just as easy to learn a fully modern programming language as it would be to learn the new versions.

Until recently, the correct rendering was "FORTRAN," but now, especially since the Fortran-90 standard, it is treated as a normal proper noun, and written "Fortran."

Business value proposition

The use of Fortran programming within organizations has decreased significantly since the 1980s as new generations of programmers coming out of universities and colleges have been taught through newer programming languages. Fortran was widely taught in the 1960s and 1970s

as an engineering programming language and thus many professional engineers, scientists, and academics of that generation consider this language to be their preferred medium for programming solutions based upon mathematical calculations. The popularity of this language during the 1970s and 1980s also grew upon the availability of library programs and packages providing easy access to statistical and numerical analysis routines, numerical control systems, and computer-aided-design systems. Subsequently many of these libraries or routines have been upgraded and may now be used with more modern languages. As a consequence many of the programmers who run Fortran programs are running systems that are weakly supported at best and with legacy versions of the packages. While the statistical abilities of the old libraries and packages may be correct, it is increasingly becoming more difficult to connect and embed these systems to other modern systems. Similarly, IT organizations will find it more and more resource-intensive to support the few Fortran programmers who wish to continue developing scientific solutions in that manner.

Summary of positive issues

Fortran is easy to program, but hard to debug. It has been in existence for over 40 years and there is a very substantial literature behind the language. Fortran formed the basis for several generations of programmers to learn to program scientific applications, and many software libraries and packages were written to leverage Fortran programs. Fortran can still be useful for small, quickly written programs that have a short lifespan and are produced by experienced programmers who do not wish to learn a new programming language.

Summary of potentially negative issues

Fortran has been superseded by newer programming languages that facilitate structured programming and object-orientated programming and offer programmers better access to operating systems functions. Owing to their essentially unstructured form, traditional Fortran programs are very difficult to debug and maintain. Fortran can be difficult for organizations to support. Many legacy Fortran programs exist.

References

- R. Barlow and A. Barnett (1998). *Computing for Scientists: Principles of Programming with Fortran 90 and C++* (New York, John Wiley and Sons).
- D. McCracken (1972). *Guide to Fortran IV Programming* (New York, John Wiley and Sons).

Associated terminology: C and C++, Java, Cobol.

Fourth generation

Foundation concept: Hardware.

Definition: First generation computers were built with vacuum-tube technology, second generation with transistors, third generation with integrated circuits, and fourth generation with large-scale integration.

Overview

The terms first generation, second generation, etc. define the technological basis upon which a computer system is built. The first automatic computing devices, starting with Babbage's Difference Engine in 1822, and culminating with Turing's Bombes in 1940, were mechanical or electromechanical in nature – they contained at least some physical moving parts – and were not programmable in the modern sense, and can not really be considered computers as the term is normally understood.

Moving parts create major problems. They are inherently slow and unreliable: overcoming friction requires considerable power consumption, and it is physically impossible for moving parts not to be

subject to continuous wear, which gradually destroys the device.

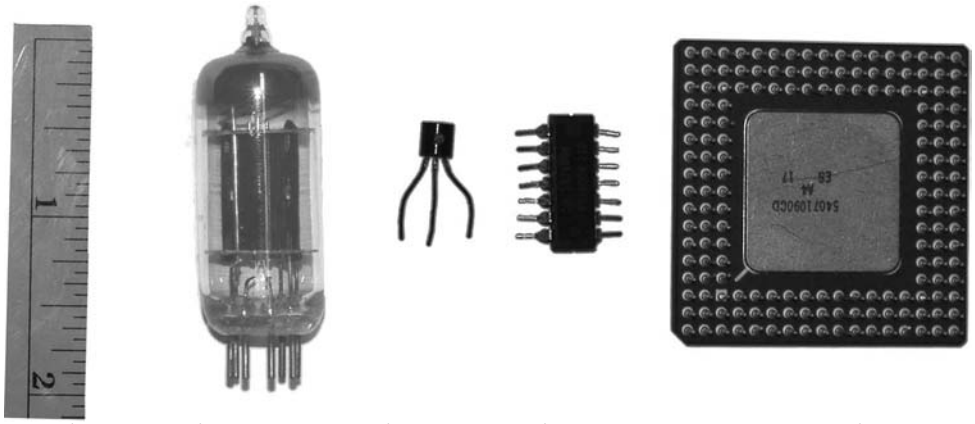
Fast reliable computing could only be built on a technology that allows machines without moving parts to be built. This seems to be a contradiction in terms, but the only thing that moves in a CPU is electricity: electrons flowing through a crystalline lattice of atoms. Other parts of a computer (disk, keyboard, mouse, CD drive) still have mechanical components, and are the only parts of a modern computer (apart from software) that are ever likely to fail.

The ability to use one electrical signal to control or switch another electrical component without any mechanical intervention was realized with the invention of the triode in 1906. Triodes and related devices, known collectively as vacuum tubes in the United States and valves in the UK, are extremely delicate and expensive. They consist of a fragile glass tube containing, in a vacuum, a small heating element that heats an electrode (the cathode) to around 4000 °F. Electrons are boiled from the surface, and fly through the vacuum to another electrode (the anode), completing an electrical circuit. A voltage applied to a third mesh electrode (the grid) interposed between the anode and cathode suppresses the flow of current, and thus allows one purely electrical signal to control another.

With their thin glass, hard vacuum, and high temperatures, vacuum tubes have a very short operational lifetime, but they can operate many thousands of times faster than any mechanical device. They were in common use for decades, but it was not until the 1930s that anyone realized that they could be applied to digital signals to perform numerical computations. This was the first generation: electronic computers with logic circuitry built from vacuum tubes. The first generation of computers started with Colossus in 1943 and ENIAC in 1946, and continued until the late 1950s.

Vacuum tubes made computers possible, but transistors made them practical. A transistor is a very robust (compared with a vacuum tube, almost indestructible), small, cheap, low-power device built from semiconductor materials. Electricity can flow through semiconductors just as electrons can flow through a vacuum, and applying a voltage correctly to a third connection can disrupt or enhance a current flowing between two primary connections, allowing one signal to control another just as with a triode. Although the physics is very different, a transistor is conceptually similar to a triode. Second generation computers used transistors in their logic circuitry where the first used vacuum tubes. This greatly improved reliability, price, and computing power. A first generation computer with perhaps 1000 vacuum tubes might be expected to run for a couple of hours before a tube burns out and has to be replaced. Significantly increasing the number of tubes would significantly decrease the mean time between failures. A second generation computer with many more transistors would usually operate reliably until some auxiliary mechanical component failed. The second generation lasted until about 1970.

A transistor is essentially a tiny piece of silicon subtly modified ("doped") in strategic places to make different kinds of semiconductor material. It is possible to take a larger piece of silicon, make a more complex pattern of modifications to it, and construct multiple transistors on a single solid component. Individual transistors have to be individually mounted and soldered onto circuit boards along with wire connectors and other parts to make a useful electronic component. An integrated circuit is a single chip of silicon with many transistors and connectors etched onto it. One small silicon chip can replace a whole circuit board *and* be manufactured in large quantities completely automatically. Third-generation computers use integrated circuits for all of their major components. This makes another significant reduction in size, power



A triode, an equivalent transistor, and two integrated circuits, one SSI, containing about 25 transistors, and one VLSI, containing about 25 000 000 transistors. The scale on the left is in inches.

consumption, cost, and fallibility. The second generation made computers practical; the third made them commonplace.

The fourth generation was a less significant step than any of the first three, and was really just a continuation of the third. Fourth generation computers apply the idea of putting whole circuits on a chip to put whole CPUs on a single chip. The fourth generation began between 1971 and 1972 when Intel introduced the 4004 and 8008, the first microprocessors. Although great improvements to integrated circuits have been made, the same basic technological idea is still in use, and we are really still in the third generation. The integrated circuits characteristic of the third generation are referred to as *SSI* (small-scale integration), and those of the fourth as *LSI* (large-scale integration) and *VLSI* (very-large-scale integration).

Generations beyond the fourth do not really exist, and the term is used by various authors to refer to hardware based on artificial intelligence, neural networks, quantum computing, or just about anything.

Business value proposition

There is no possible advantage to using a first or second generation computer except in the context of a museum. The latest generation simply represents the latest technology and quickly renders previous generations obsolete. Of course, it might not be wise to jump to the next generation as soon as it is announced (or claimed), but, once new technology has become established and is known to be reliable, the old effectively ceases to exist, leaving businesses with no options to worry about.

FTP (file transfer protocol)

Foundation concepts: Network, File.
Definition: A network data-transfer application.

Overview

FTP is a simple but effective and fairly efficient method for transferring files of any kind between networked computers. It is a typical client–server system.

Although FTP is almost universal, and is very simple to use, it does require some foresight. One of the two computers involved in the data transfer, either the receiver or the transmitter, must have an FTP server installed on it. FTP clients are ubiquitous – every operating system provides at least a basic one – and they require

no special setup, but FTP servers are not always provided for free, and need to be installed and configured correctly. Installation and configuration is just a few minutes' work for the right personnel, but must be done correctly. A mis-configured FTP server can leave every piece of data on the computer open to the entire internet.

In normal use, a user wishing to transfer data will start up the FTP client software and tell it to connect to the desired server system. The user provides a user-name and password, and, if they are accepted by the server, files and folders may be transferred in either direction, often with a single click or drag-and-drop operation. Server administrators may control exactly which files and folders any given user or group of users may access, and what kind of accesses (read, modify, delete) they may make.

FTP also has an anonymous mode, in which certain files are made accessible to all, without the need for a user-name or password. This provides a very convenient means for making files publicly available, but is also a possible security flaw: accidentally allowing anonymous FTP circumvents file protections. Most web browsers understand FTP, and provide a very intuitive interface for downloading files. For large amounts of data, FTP is superior to HTTP, the default web protocol.

FTP can be problematical for firewalls, since every FTP transfer requires the use of an arbitrarily selected internet port, so simply leaving "the FTP port" unblocked is not enough. Modern firewalls are constructed with complete knowledge of FTP and other commonly used protocols, and are usually able to handle the situation, albeit with some complexity.

Business value proposition

FTP is a simple, efficient mechanism for data transfer between two computers connected over a network. The approach allows large amounts of data to be transferred easily and effectively. FTP's almost universal availability on computers makes this approach useful for data transfer anywhere a network connection is available.

Summary of positive issues

FTP is a cheap, efficient, and widely available method for data transfer.

Summary of potentially negative issues

It requires care in configuration. Security issues in several areas need to be addressed, including the use of ports used to communicate through, the firewall configuration, and the data files that are allowed to be accessed.

References

- W. Stevens (1994). *TCP/IP Illustrated* (New York, Addison-Wesley).
- http://www.ietf.org/rfc/rfc0959.txt?number=959.

Associated terminology: File server, Server–client, Electronic data interchange, X.12.

Functional programming

Foundation concepts: Programming language, Formal methods.

Definition: A style of programming in which computations are expressed as pure mathematical functions that can be directly executed.

Overview

Functional programming is a widely misunderstood concept. The description is simple – pure mathematical functions are used to express computations (programs) – but the true significance of "pure mathematical" is often missed. A function in the mathematical sense is not what most programmers think of as a function. A mathematical function is a fixed mapping from each of a set of possible inputs to a corresponding output. The same function applied to the same inputs must always deliver the same results, regardless of

circumstances. In fact, the idea of "circumstances" does not mean anything in this setting; mathematics does not vary according to context.

This very basic notion of requiring that the same function applied to the same inputs always produces the same results is essential to all of mathematical reasoning. The standards of proof simply do not work if you can write the same thing twice but have it represent a different value each time. If this absolute constancy is applied to the design of programming languages, it becomes possible to produce perfect proofs of a program's correctness. New techniques of program development become possible, in which a statement of the problem to be solved is simply *transformed* into a solution to that problem.

The difficulty is that standard programming languages do not behave at all like this. The vast majority of programmers would consider it impossible to write a program to perform any useful task if it is not allowed to change anything. True functional programming does not permit variables, loops, "print statements," or any of a multitude of features normally thought of as essential to programming. In order to take advantage of the vast benefits that functional programming has to offer, programmers have to almost relearn their trade right from the beginning. That is not an easy, or a cheap, thing to do.

There are some popular programming languages that are often called functional, when they really are not. Lisp, the workhorse of artificial intelligence, is very frequently thought of as a functional language because in Lisp everything *looks like* a function. Appearances are irrelevant; Lisp is not a mathematically functional language, and hence does not provide the benefits of one. That is not to say that it is a bad language; Lisp is very useful, it is just not functional.

Many real functional languages are available, and it has occasionally been one of the major areas of computer science research. Popular implementations that either are purely functional or at least support mathematically functional programming include LispKit, KRC (Kent Recursive Calculator), Miranda, Haskell, AFL (A Functional Language), FP, Hope, and Clean, but there are many more. A theoretical construct known as *Monads* is a topic of current research, attempting to make the benefits of functional programming available to more traditionally oriented programmers.

Business value proposition

Functional programming has a vast potential for the development of mathematically correct programs. Functional programming constructs also help programmers to create solutions in such a way that they are not constrained by arcane rules of procedural programming. Unfortunately, many programmers are unaware of this branch of computer science and hence the use of this style of programming in the "real world" has been limited.

It should be noted that the amount of "lines of code" required to perform an operation in a functional language is frequently very small compared with that for a procedural language, and the functions themselves can also be easy to read with some training. The style of programming has the potential for providing cleaner, smaller (in terms of amount of code), partially self-documenting programs, which reduces the overhead on the developer. However, when programmer productivity is measured in terms of lines of code delivered, a very distorted picture is given.

Summary of positive issues

Functional programming is mathematically pure and is open to undisputable proof. A significant academic literature has developed on the topic. Many free functional programming languages are available. The programs themselves can be

transformed and reasoned about mathematically.

Summary of potentially negative issues

Functional programming needs an awareness and knowledge of pure mathematics. The approach is not widely understood or known amongst programmers. The interpreters are often relatively slow at executing the code.

References

- S. Peyton-Jones (1997). *Implementing Functional Programming* (Englewood Cliffs, NJ, Prentice-Hall).
- J. Darlington (1982). *Functional Programming and Its Applications* (Cambridge, Cambridge University Press).
- R. Bird (1998). *Introduction to Functional Programming* (Boston, MA, Pearson).
- P. Hudak (2000). *The Haskell School of Expression* (Cambridge, Cambridge University Press).

Associated terminology: Logic programming, Formal methods, Software development.

Fuzzy logic

Foundation concept: Knowledge-based systems.
Definition: A system of logical reasoning that handles imprecise knowledge or information.

Overview

Fuzzy logic is a reasoning system that allows truth values other than "yes" and "no." The validity of a statement is expressed as a number somewhere between 0.0 (for completely false) and 1.0 (for completely true). For example, the truth value of "two plus two is four" would probably be exactly 1.0, whereas the truth value of "the Empire State Building is tall" might be 0.85. This is not because we are uncertain about the height of the Empire State Building, and it does not mean that it is *probably* tall. It is because the idea of tallness is not precisely defined, it is a fuzzy concept. The 0.85 represents the judgment that it is *quite* tall. By most standards the Empire State Building is tall, as when compared with people, other skyscrapers, or red ants, but it is not tall by all standards: when compared with mountains or thunderclouds, it is distinctly short.

The use of fuzzy logic allows automatic decision-making processes to progress when an exact algorithm can not be found, and it is generally considered to be a technique of artificial intelligence and expert systems working in the domain of human descriptions. For example, a search for "big orange cats" would be hard to code with an exact algorithm because both "big" and "orange" are fuzzy human-oriented descriptions. It would be pointless to ask the user for further details, exactly specifying how big in terms of inches or pounds, and it would be absurd to attempt to quantify how orange the cat must be. A fuzzy logic system would allow "big" to be determined as a fuzzy combination of "long" and "heavy," accepting candidates that are quite long and quite heavy, together with those that are very long but only moderately heavy, and those that are very heavy but only moderately long, thus encapsulating all reasonable meanings of "big." It could then similarly filter all candidate cats, having reduced requirements for the degree of orangeness for those that are exceptionally big, and reduced requirements for the degree of bigness for those that are startlingly orange.

Fuzzy logic is very difficult to work with effectively. The assignment of levels of tallness, bigness, and orangeness on the basis of measurable quantities is entirely arbitrary and a matter for the system designer to decide. This makes success almost impossible because all users will have different ideas of how these terms should be defined. Then there is the problem of how to

combine fuzzy values: if a cat is big to the tune of 0.74, and orange to the tune of 0.9, how should those numbers be combined to give a single measure of how "big orange" it is? Again, there are no clear answers; it is for the system designer to decide, and for the users to inevitably disagree.

For these reasons, fuzzy logic has not been a very commonly used tool. At first sight it seems to be an obvious and good idea, but it is very resistant to encoding in the precise language of computer logic.

Fuzzy logic was introduced by Lofti Zadeh of the University of California in Berkeley in 1965. He and others have developed a properly mathematically rigorous basis for fuzzy logic, which is known as possibility theory. The notation and concepts of possibility theory can easily be confused with probability theory, but it is a completely different thing. Probability theory is entirely uncontroversial and admits no imprecision, only uncertainty. Fuzzy logic and standard probability theory may be applied to similar computational situations, but have very different semantics. Saying that the probability of X being tall is 0.85 means that we don't really know whether X is tall or not; it probably is, but we can't be sure. Saying that the fuzzy measure of X's tallness is 0.85 means that we are absolutely certain that it is quite tall. Probability deals with uncertainty over well-defined attributes; fuzzy logic deals with certainties about ill-defined attributes. Exactly how connectives like "and" and "or" combine probability values is well known; how they combine fuzzy logic values is inherently unknowable.

Business value proposition

Fuzzy logic was originally developed in the 1960s by Lofti Zadeh as a mechanism through which "fuzzy" problems could be considered. While probability theory can be applied to a variety of problems, it is not suited to all, and it requires that the problem solver has sufficient historical data to make a determination. Fuzzy logic allows software engineers to create programs that work for a variety of situations based upon imprecise knowledge of inputs, the state of the system, and even the desired results.

Fuzzy logic technologies have been incorporated into a wide variety of processes and products, including those found in automobiles, e.g., cruise-control systems, traction-control systems, anti-lock systems, and cornering systems; household goods such as toasters, microwave ovens, and refrigerators; and entertainment products such as video cameras for automatic exposure and focus control.

Summary of positive issues

Fuzzy logic allows programmers to use truth systems that are non-binary in nature. It allows determination of outcomes when probability theory is not applicable. Fuzzy logic is a well-researched academic discipline and can be applied to a wide variety of problems and processes.

Summary of potentially negative issues

Fuzzy logic mathematics can become complex and it requires some special training. Fuzzy logic is difficult to work with effectively. The combination of fuzzy values in logic is difficult to determine in many situations.

Reference

- G. Klir and B. Yuan (1995). *Fuzzy Sets and Fuzzy Logic* (Boston, MA, Pearson).

Associated terminology: Artificial intelligence.

Global positioning system

Definition: Also known as NavStar GPS, an electronic device that may rapidly and accurately determine its own geographical location by receiving signals from a network of orbiting satellites.

Overview

There are a few dozen special-purpose satellites, launched and maintained by the US military, in constant Earth orbit. They contain extremely accurate clocks, programmed knowledge of their own orbits (which is constantly corrected by signals from ground stations), and high-frequency radio transmitters. These satellites transmit a constant stream of data, which describes the exact time and their position, and can easily be picked up by terrestrial receivers.

A GPS receiver listens to the stream of data from a number of these satellites, and from it, using relatively simple geometry, can calculate its own exact position (longitude, latitude, and elevation) to an accuracy of just a few feet. These receivers can be made very small (around one square inch) and moderately cheaply (tens of dollars). Their primary use is as an aid to navigation (in aeroplanes, in smart guided missiles, and in hand units for use by people), but they may also be embedded into larger systems. By comparing a sequence of positions, a GPS unit may also determine its own speed and direction of movement, and the relevant portion of a digitized map may be automatically displayed.

Coverage is global, but does require an almost unobstructed "view" of the sky. Signals do not penetrate anything but the flimsiest of structures, and can even be obscured by trees overhead. GPS receivers built into vehicles need to be in a relatively open position. A single GPS receiver can determine position, but not orientation (i.e., it can not act as a compass); two receivers working together can determine orientation by comparing their relative positions, but would need to be some tens of feet apart to do so.

Because GPS can be (and is) used to guide weaponry, the US military has retained the ability to deliberately degrade the transmitted signal (make it less accurate), or turn it off completely, during emergencies. Naturally, military receivers are able to overcome the degradation, and give positions much more accurately than do models available to civilians. Even during normal operation, military receivers produce a much more accurate fix than do civilian receivers.

Business value proposition

The United States-based NavStar GPS and the European Galileo GPS, which is scheduled to be available from 2008, are technologies used by civilian organizations in a variety of ways. Typical uses include automobile and maritime guidance systems, vehicle-tracking systems (some United States-based car-rental companies prevent their vehicles from being restarted when driven into Mexico), surveying systems, and automobile safety systems (crash locations can be automatically reported to emergency services through GPS and transponder technologies).

Summary of positive issues

GPS systems are inexpensive, accurate, and can locate a position globally. From 2008 two versions of the system will be available.

Summary of potentially negative issues

The GPS could be downgraded in times of political tension. As a satellite-based system it is subject to solar radiation and other solar threats. The GPS works only outdoors where there is a clear "line of sight" with the satellite; even tunnels and trees affect coverage.

Reference

- A. El-Rabbany (2002). *Introduction to GPS: The Global Positioning System* (New York, Artech House).

Groupware

Foundation concept: Decision support system.
Definition: Groupware is a software platform that enables a team to undertake a joint task and provides support.

Overview

Groupware provides a software platform that allows groups to collaborate on a task or project. It is sometimes termed *Computer-supported cooperative work* (CSCW) or *Computer-mediated communications*. Groupware is a mechanism that Johansen categorized as being composed of two parameters. The first is location: are the team members located in the same place or not? The second is temporal: are the team members working on a task simultaneously or not? For example, video conferencing is a simultaneous process wherein members participating are at different locations, whereas text messaging via cellular phones is asynchronous for participants at different locations.

Groupware may be used to support synchronous tasks, for example the production of an engine component for an automobile; in such a situation a team may be located in a single central design studio with each team member working on a separate task. One team member could be working on the interface between the components, while another team member could be developing the packaging. In this situation product-versioning and document-management software is also a useful component to support the team and task development.

Groupware not only supports the process task (such as the virtual modeling of the CAD for an engine component), but also needs to provide project management support in the form of email or instant messaging, calendar management functions, and so on.

Another popular form of groupware is multi-user computer gaming, in which users share a virtual memory space to experience the synchronous interaction associated with the game. Although this is an entirely recreational use, the concepts of multi-user interactive games have been transported into the business world in the form of virtual meetings.

Business value proposition

Groupware has the potential to help organizations utilize their workforce in a more productive manner in that team members may be at different locations and work in shifts, but can still work together on a project. Groupware enables team members to work in asynchronous mode, which, when supported by knowledge-management software, can produce high degrees of efficiency.

Summary of positive issues

Groupware is available and supported by consultants and vendors, and can be used to promote greater flexibility within the organization's structure. Groupware can be used to monitor activities of team members and track human-resource expenditure.

Summary of potentially negative issues

Companies have been hesitant to implement groupware since it requires organizational and process changes. When group members operate remotely through groupware, their activities may need human monitoring to ensure that performance levels are maintained.

References

- T. Hall (2006). "Intelligence community collaboration" (Washington, DC,

Solutions Office for Advanced Analytical Tools, Central Intelligence Agency), available online only, from http://collaboration.mitre.org/prail/IC_Collaboration_Baseline_Study_Final_Report/toc.htm.

- R. Johansen, J. Charles, R. Mittman, and P. Saffo (1988). *Groupware: Computer Support for Business Teams* (New York, Free Press).

Associated terminology: Knowledge management.

Hacker

Foundation concept: Security.
Definition:

1. An undisciplined, unprofessional programmer.
2. Someone who attempts to break into other computer systems.

Overview

1. Originally, a hacker was a programmer who worked without any clear plan, but would just type something in and "hack it around" until it seemed to work. A hacker would often be very well informed about the particular programming language and the technical details of the operating system they were using, and would know a wide variety of surprising tricks, but would usually not be very well educated in the science and art of programming.

However, hackers always considered themselves to be the ultimate experts, and their facility with interesting tricks and their propensity for constructing software incomprehensible to others tended to reinforce this view, and "hacker" gradually became a mostly positive term. It is from these roots that the second definition evolved.

2. In modern usage, "hacker" has come to mean only one thing: a person who attempts to break into other computer systems. The motivation may be anything from basic pride or vanity to theft, sabotage, espionage, or revenge. The results may be anything from a simple web-site posting to prove that a break-in was successful, to complete destruction of corporate data and modifications to essential records or access to confidential data that is never detected. The methods range from simply guessing at passwords (a surprisingly successful technique), through befriending careless or disgruntled employees, to transmitting targeted viruses, Trojan horses, and worms. A very common means of illicit entry is to exploit insecurities that already exist in the installation's software infrastructure (operating system and applications) that were included by careless developers.

On the basis that "it takes a thief to catch a thief," ex-hackers (or supposedly "ex," at least) can be valuable members of a security team. Who would be better able to guard against break-ins than the very person who had previously devoted his life to performing them?

Business value proposition

There is very little business value associated with the first definition of the term, because systems that were generated by hackers do not typically conform to corporate programming methodologies or standards. This makes them difficult to read, interpret, and alter; and efforts to do so expend large amounts of resources. These systems are typically problematic and legacy in nature, although companies that still employ programmers with this type of mentality toward programming will continue to generate poor-quality systems and code until best-practice standards and approaches to software design are enforced. A traditional hacker may be able to produce a mock-up of software somewhat quickly, but totally unreliable software is of very little value.

The modern form of hackers who break into systems are a corporate liability. The employment of a hacker unbeknownst to the organization can lead to severe problems for the company because intellectual property may be stolen and released to the public, the security and integrity of the corporation may very well be compromised, and the company may be used as a base for illegal activities. Companies must undertake background vetting of all systems personnel prior to the assignment of a security clearance.

Organizations need to put in place security precautions to prevent all unauthorized systems access. This includes access

attempts from hackers both internal and external to the organization. The employment of a chief security officer is critical in large organizations, and the use of skilled specialists to undertake frequent security assessments is vital in order to maintain objectivity.

Summary of positive issues

Hacking as a style of programming is less common in organizations employing strict programming standards and methodologies.

The unauthorized hacking into systems can be defended against through the employment of security measures, such as mandating well-chosen passwords, installing a reliable firewall, closing all accounts immediately an employee leaves employment at the firm, and installing the latest security patches from vendors.

Summary of potentially negative issues

Hacking as a style of programming is wasteful of resources, difficult to maintain, and potentially dangerous because the system produced might not perform as required under all circumstances.

The issue of hackers who break into corporate systems to access unauthorized information is extremely problematic, and can be the cause of significant losses and liabilities. Hackers internal to the company can reveal intellectual property and cause liability issues for the company.

References

- S. Harris, M. Lester, A. Harper, C. Eagle, and J. Ness (2004). *Gray Hat Hacking: The Ethical Hacker's Handbook* (New York, McGraw-Hill).
- K. Mitnick and W. Simon (2005). *The Art of Intrusion* (New York, John Wiley and Sons).

Associated terminology: Virus, Trojan horse, Cracker.

Hardware, software, firmware,

Definitions and overview

Hardware is the physical part of a computer: the circuit boards, the silicon chips, the wires, the boxes, the disks, etc. Hardware is the parts that can be seen or touched.

Software is the more ephemeral components that exist only as electrical impulses in conductors, magnetic fields or optical distortions in storage media, or electrical charges in memory capacitors. Software is the programs and applications that run on a computer and make it useful. The term software is also frequently used to include the passive data stored on computers: images, databases, email messages, and so on.

Firmware is the very-low-level software, often safely encapsulated in immutable ROM (read-only-memory), which is so intimately connected with the operation of the hardware that it almost seems to be part of the hardware itself. The best known example is the BIOS: very basic software responsible for starting up a computer when it is first turned on, before the operating system has started up.

A **program** is an active piece of software, the encoding of an algorithm or computational procedure in a form suitable for execution by the computer. The occasionally heard term "software program" is redundant. In American English, the word "program" is spelled the same way in all of its senses. In British English, the spelling "program" is now always used when "software" is meant, the spelling "programme" is reserved for the other senses, e.g., "programme of entertainment."

Application is simply an alternative word for program. Often, "application" is reserved for major software products, but the terms are generally taken to be synonymous.

References

- D. Groth (2003). *A+ Complete* (Hoboken, NJ, Sybex–John Wiley and Sons).
- R. Thompson and B. Thompson (2003). *PC Hardware in a Nutshell* (Sebastopol, CA, O'Reilly Press).

Health Insurance Portability and

Foundation concept: Security.

Definition: The Health Insurance Portability and Accountability Act of 1996 (HIPAA) protects the privacy of personal medical records. It aims to improve the portability and continuity of health-insurance coverage and to provide a framework for obtaining, managing, and disseminating medical or personal information, termed electronic protected health information (EPHI).

Overview

The HIPAA legislation consists of five titles: Title I covers "Health care access, portability and renewability"; Title II covers "Preventing health care fraud and abuse; administrative simplification; medical liability reform"; Title III covers "Tax related health provisions"; Title IV covers "Application and enforcement of group health plan requirements"; and Title V covers "Revenue offsets."

The US Department of Health and Human Services issued a special rule known as the "Standards for Privacy of Individually Identifiable Health Information" ("Privacy Rule") to help healthcare providers and businesses implement and manage the HIPAA. The privacy rule focuses upon the protection of personal healthcare information and covers the standards acceptable for the electronic exchange, privacy, and security of health information.

Business value proposition

The HIPAA provides a comprehensive framework for healthcare providers and the organizations that interact with those providers. The act impacts information systems in a variety of areas, including electronic-transaction standards and information-security standards.

The HIPAA has mandated standards for "administrative and financial health care transactions" based upon ANSI ASC X.12N, Version 4010, except for retail pharmacy transactions, which will use the National Council for Prescription Drug Programs (NCPDP) Telecommunications Standards Format Version 5.1. The standards are monitored by six designated standards-maintenance organizations (DSMOs), namely ASC Committee X.12, The Dental Content Committee, Health Level Seven, the NCPDP, the National Uniform Billing Committee, and the National Uniform Claim Committee.

The HIPAA details the obligations of entities operating under the Final Rule (45 CFR Parts 160, 162, and 164) of the act including administrative safeguards on

(1) security-management processes;
(2) assignment of security responsibility;
(3) workforce security;
(4) information-access management;
(5) security awareness and training;
(6) the implementation of security-incident procedures;
(7) contingency-planning processes;
(8) evaluation policies; and
(9) business-associate contracts.

Physical safeguards are also proposed, including

(1) assigned security responsibilities;
(2) media controls,
(3) physical-access controls;
(4) workstation-use policies;
(5) workstation-security location; and
(6) security-awareness training.

The following technical safeguards have been proposed:

(1) access controls;
(2) authorization control;
(3) data and entity authentication;
(4) integrity controls;
(5) message authentication;
(6) encryption;
(7) alarms;
(8) audit trails; and
(9) event reporting.

Summary of positive issues

The act provides a comprehensive framework for healthcare organizations. The Department of Health and Social Security has set compliance dates for each part of the act.

Summary of potentially negative issues

Compliance with the HIPAA standards requires significant expenditure of human and capital resources.

References

- Public Law 104–191, August 21, 1996, Health Insurance Portability and Accountability Act of 1996.
- D. Solove and M. Rotenberg (2003). *Information Privacy Law* (New York, Apsen).
- M. Rotenberg (2003). *Privacy Law Sourcebook 2003* (Washington, DC, Electronic Privacy Information Center).

Associated terminology: Security, Encryption, Law cross-reference.

Host, host name, hosting

Definitions:

Host: Any computer.

Host name: A unique identifier for a networked computer.

Hosting: A third-party service providing computer equipment, software, maintenance, and network access.

Overview

Host: In technical descriptions of internet protocols and related documents, the word *Host* is used to refer to any kind of computer or other device that can perform the same tasks, such as "smart" telephones and other appliances, intelligent routers, and so on. This is simply to provide a single cover-all term and avoid clumsy phrasing; it has no other significance.

Host name: The name or address of an internet-accessible computer, in user-friendly form. Often called a *Web address*. A host name consists of a *Domain name* (q.v.) with optional prefixes to specify an individual host. For example, "www.corporation.com" and "sales.eastern.company.co.uk" are both host names, not domain names; the domain names are "corporation.com" and "company.co.uk," respectively.

Hosting: When an organization or individual wishes to have a web presence, the associated costs can be significant. If only light access is expected, a simple DSL, ISDN, or cable-modem service from a telecommunications or cable provider, connected to a desktop computer in the office or home, may be sufficient. If exceptionally light access is expected, the expense of maintaining an "always-on" internet connection with the static IP address required for reliable public access may make even this too expensive. If even moderately heavy access is expected, such a simple system will probably not suffice. In both of these cases, a *Hosting service* may be the solution.

A hosting service is a company that already has a physical location with a fast internet connection, uninterruptible power supplies, some physical security, computers ready for use, and support staff to keep them running. The services they provide range from a small share of space and bandwidth on an existing web server to support a personal web site, all the way up to 24-hour maintenance on a group of servers

dedicated to providing a range of network services for a single customer.

Business value proposition

A web-hosting service is an attractive proposition for companies that wish to relieve themselves of the burden of managing their own web server. The ability to outsource this function allows organizations to focus upon their core web activities such as managing the site content, developing the applications, and managing the customer-service relationships, logistics, and other aspects of their business. Web hosting services can also be used as backup service providers should a disaster occur at the corporate site.

Summary of positive issues

Web hosting provides a relatively low-cost, highly reliable solution to managing web servers and an alternative means of providing web service should in-house systems fail.

Summary of potentially negative issues

Naturally, prospective customers of a hosting service should carefully consider the guaranteed minimum levels of service and security specified in contracts, and be fully aware of the costs associated with higher-than-anticipated bandwidth use.

Reference

- D. Kaye (2001). *Strategies for Web Hosting and Managed Services* (New York, John Wiley and Sons).

Associated terminology: Web services, ISP.

Human–computer interaction

Definition: Human–computer interaction is the study, design, and evaluation of interactive computing environments as used by human subjects in order to accomplish a process or task.

Overview

Human–computer interaction (HCI) has been a part of computing in a variety of ways since computer applications transitioned from being batch-mode transaction processing and became interactive. HCI as a discipline has two main dimensions: *Human factors*, which considers the problems humans encounter with information presentation (e.g., does a pilot prefer dials or digital readouts for their aircraft instruments); and *Ergonomics*, which considers the issues surrounding the design of objects for human use (e.g., keyboard design to reduce the risk of repetitive-stress injury).

The study and development of HCI accelerated in the 1970s when researchers at Xerox PARC devised the first Windows-based GUI (graphical user interface) and invented the mouse as an interface device. Other researchers pioneered interface devices such as the tracking ball, the joystick, touch-screen interfaces, and the light pen. These mechanisms allowed greater flexibility in the way that interfaces could be constructed and the means by which users could work with their systems.

A chief proponent of HCI development has been the military, whose systems have become progressively more interactive. Military simulations require the use of virtual-reality models and flight simulators that allow participants to practice within a controlled environment. In the simulators, as on their aircraft, pilots wear a "heads-up" display on which the flight and weapons data is projected into their field of view; this form of HCI is aimed at improving the reaction times and performance of the pilots.

The computing environment used by millions of people every day is frequently just taken for granted, but there are many devices and mechanisms through which users can and do interact with computers, again usually without thought, and this ease of use is a direct result of HCI research. For example, many automobile controls are

actually computer interfaces, including the "drive-by-wire" accelerator and braking systems, the hands-free (Bluetooth) mobile-phone operation, the navigation system, heating and entertainment systems, and the steering-wheel "paddles" for gear shifting.

Business value proposition

The value of HCI research is to be found in a variety of areas, including the safer and more accurate use of computer systems (e.g., automotive, military). Correct assessment of HCI issues can result in a more productive workplace where computer users are not taking time off for computer-use-related medical problems such as back pain, carpal tunnel syndrome, and eye strain.

Summary of positive issues

HCI is an area of active research in academic and commercial organizations. HCI has an extensive literature. HCI can lead to more productive workforces.

Summary of potentially negative issues

The incorporation of HCI research into a standard human–machine interaction may require significant expenditure of resources, but there is a potential for a high return on this investment.

Reference

- J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey (1994). *Human–Computer Interaction: Concepts and Design* (New York, Addison-Wesley).

Associated terminology: Virtual reality.

Hypertext, HTML

Foundation concept: World Wide Web.
Definition: Text enhanced with additional features such as automatic formatting, images, and links to other documents

Overview

Plain-text files are simply sequences of characters and line-breaks exactly as they could have been produced on an old-fashioned manual typewriter. When computers were strictly technical tools and computer interconnectivity was low-bandwidth if it existed at all, all documents were either plain text or in some application-specific format. Plain text was perfectly adequate for the intended purposes, but is of little use in the world of desktop publishing and graphic-conscious web pages.

When computer typesetting and desktop publishing first appeared, macro languages became popular. The premise is that text is typed exactly as normal, but, whenever some special action is required, special symbolic tags are inserted into the text. When the author is working on the text, these tags are nothing special; they just appear as strange symbols in the text, and can be edited along with everything else. When the document is specially processed for printing, the tags are not displayed at all; instead, they control various aspects of the document's appearance.

One of the most successful applications of this kind was the TeX/LaTeX system from Donald Knuth and Leslie Lamport. It was observed that the character "\," which appears on all computer keyboards, is never used in normal writing, so it was taken as a special flag symbol. In the LaTeX system, text is taken exactly as it is typed, until a "\" sign is encountered, at which point the following text component is taken as a special typesetting command. For example, a LaTeX document may contain this text:

```
the shop-keeper insisted it was a
  \it heffalump,\rm
and cost \pounds 7.50; this was
less than \frac{3}{4} of the
  price I had
seen elsewhere, but the coloration
  was
rather pass\'{e}.
```

It is quite easy to read as it appears, which is very helpful when editing, especially with the knowledge that "`\it`" is a command to start using the italic typeface, "`\rm`" is a command to return to the normal roman typeface, "`\pounds`" represents the British currency symbol, and so on. After processing, the above text would be rendered on a printer as

> the shop-keeper insisted it was a *heffalump*, and cost £7.50; this was less than ¾ of the price I had seen elsewhere, but the coloration was rather passé.

Hypertext systems are really just an extension of this basic style. By far the most common in current use is HTML, the HyperText Mark-up Language, the standard for which is maintained by the W3C organization. HTML is the universal language for web pages, and, although various manufacturers' browsers make significant variations on the details of the standard, basic HTML is universally understood by all web browsers.

Although HTML has a slightly different intent from LaTeX, it is very similar in appearance, and gives the same advantages: authors may work with simple plain-text files but produce very sophisticated layouts. The example above would appear thus:

```
the shop-keeper insisted it was
a <em>heffalump,</em> and cost
&pound;7.50; this was less than
&frac34; of the price I had seen
elsewhere, but the coloration was
rather pass&eacute;.
```

The markup tags in HTML are intended to indicate the purpose of the text, rather than to exactly control its appearance. A clear example is the `<em>` tag used above; this indicates that the affected text is to be emphasized in some way. Most web browsers will provide emphasis by using italics, but that is not a requirement. In retaining some flexibility in how documents are to be displayed, HTML is able to adapt to a great variety of conditions. A user may have reduced the web-browser window to a long thin stripe to keep it out of the way. HTML should still be able to fill it with text and keep it readable, whereas a more exacting language like LaTeX would insist on an exact layout for a specific page size and would not leave browsers free to adapt.

This great flexibility in HTML has led to a great deal of dissatisfaction amongst the authors of web content, who often have a very fixed idea of exactly how their web pages should appear, with carefully planned positions for graphical elements and perfectly selected artistic fonts. It is often possible to wrest complete control from the browser by using newer HTML extensions and such features as CSS (cascading style sheets) and JavaScript controls. The result can be most unfortunate, producing web pages that can be viewed only at one particular size; there are many commercial sites that are completely unprintable because they have been overengineered for the traditional short-but-wide monitor configuration. HTML should be used to provide general layout guidance, not exact typesetting instructions. Hypertext is not necessarily tied to HTML, but, simply because there are currently no competing systems with such universal support from web browsers, and such universal and free accessibility to authors, HTML does currently define hypertext.

The most essential feature of hypertext is not its ability to specify the appearance of text with markup elements, not even its ability to include full graphics integrally in the display of a document; these features merely reproduce what is available in books. The definitive feature of hypertext is the ability to actively link documents together. Whilst reading one document, a person may decide to follow a reference or look up an unfamiliar word or phrase. With a simple click of the mouse, the appropriate document appears on the screen along

with the original. The viewer may read the new document, then return to the original, or read the two together, or follow up on another reference in a limitless chain. The immediate and concurrent access to linked documents without the requirement for any searching is what puts the "hyper" into "hypertext."

The support for connected documents, or *Hyperlinks*, provided by HTML is surprisingly primitive. The author of a document must decide exactly which other documents should be linked with any given section of his or her composition, and explicitly add markup tags giving the location of the document as a URL (universal resource locator, q.v.). There is no mechanism for specifying a general look-up rule for all words in a document, nor is there any mechanism for ensuring that the linked document has not changed, and is still relevant. What we have today is barely more sophisticated than the Memex system invented by Vannevar Bush (but never built) in 1932; he envisioned a library of documents on microfilm physically linked together with fine threads, and a special machine to display those documents and follow threads when the user pulls a lever.

The World Wide Web as we know it today is for the most part a widely distributed collection of hypertext documents. Most are written in HTML, some still in plain text, and some use proprietary systems (the most popular of which is Acrobat), for which viewers are available for free, but composition requires commercially available software. Surprisingly, there are no LaTeX viewers available that can be integrated with the standard web browsers.

Business value proposition

HTML is universal in its use as the base for hypertext web documents. HTML is a non-proprietary format developed through the W3C.org consortium and the HTML Working Group. HTML continues to evolve, with a working draft of the Extensible HyperText Markup Language (XHTML) version 2.0 having been released on May 27, 2005. XHTML (HTML written in XML) advances certain aspects of earlier versions of HTML and is backwards compatible, such that all older HTML documents continue to display correctly on newer browsers.

Summary of positive issues

HTML is an open-source, non-proprietary system. Basic HTML is easy to program, flexible, universally used in hypertext development, and understood by all browsers. HTML continues to evolve through the activities of W3C.org. HTML provides a strong basis for web-site development and tools are available to help developers improve their productivity.

Summary of potentially negative issues

HTML has become by default the standard vehicle for hypertext presentation. Organizations need to deploy resources to understand it fully and understand its limitations. This may be in the form of training and/or being active in the W3C.org HTML Working Group. The flexibility of HTML can be problematic to developers wishing to develop a web page with fixed dimensions. HTML is the dominant web development language and will remain so for the foreseeable future.

References

- C. Musciano and B. Kennedy (2002). *HTML and XHTML: The Definitive Guide* (Sebastopol, CA, O'Reilly Press).
- P. Gralla (2004). *How The Internet Works* (Indianapolis, IN, Que).
- L. Lamport (1986). *LaTeX User's Guide & Reference Manual* (New York, Addison-Wesley).
- V. Bush (1945). "As We May Think," *Atlantic Monthly*, July.
- http://www.w3.org/TR/REC-html40/.

ICANN (Internet Corporation for

Foundation concept: Domain name.
Definition: The non-profit corporation delegated responsibility for the allocation of IP addresses and other internet-related identifiers by the US Department of Commerce.

Overview

IP addresses and domain names are critical to the operation of the internet, and at the top level must be managed by a recognized authority. The US Department of Commerce somehow became the international authority for such things, and delegates that authority to the Internet Corporation for Assigned Names and Numbers (ICANN). It is a non-profit corporation, with an international board, and has subsumed the former responsibilities of the Internet Assigned Numbers Authority (IANA).

ICANN usually allocates IP addresses in large groups to regional authorities, who then allocate them individually or in smaller groups to ISPs and other organizations. ICANN is also responsible for the overall management of DNS, the Domain Name Service, and creates the top-level domains (such as ".com" and ".org") and the national domains (such as ".uk" and ".nu").

ICANN is often blamed, completely unfairly, for the fact that IP addresses are running out, and organizations in the developing world might not be getting the share they would like. They are also sometimes blamed, perhaps less unfairly, for the fact that prices for registration are much higher than they once were, and this is seen as unreasonably favoring the old established organizations over new ones. Most organizations have no direct dealings with ICANN, since domain-name registration and IP address allocation is usually handled by intermediaries such as ISPs and hosting services.

Business value proposition

ICANN is the ultimate authority in the allocation of domain names and IP addresses, and in the creation of top-level domains. There are currently seven top-level domain names aside from the national domains; however, there have been proposals to create new top-level domain names, including ".asia," ".cat," ".jobs," ".mail," ".mobi," ".post," ".tel," ".travel," and ".xxx." ICANN has requested public comments on these proposals.

Most organizations will have no direct dealings with ICANN, since domain name registration and IP address allocation is usually handled by intermediaries such as ISPs and hosting services.

Summary of positive issues

ICANN controls the top-level domain names. Registration of domain names is well organized and they can be acquired through ICANN-authorized companies such as ISPs.

Summary of potentially negative issues

The top-level domain names have been fixed and expansion is slow to occur. Many domain names have already been taken. Some domain name categories are less attractive to corporations attempting to create their brand than others.

References

- http://www.icann.org.
- ICANN, 4676 Admiralty Way, Suite 330, Marina del Rey, CA 90292, USA.

Associated terminology: Host, Domain name, Internet, World Wide Web.

IEEE (Institute of Electrical and

Definition: A professional organization for individuals involved in all areas of electronics, particularly computers.

Overview

The IEEE, the Institute of Electrical and Electronic Engineers, is a major international society, probably the largest professional organization in the world, with over 300 000 members. It is concerned with all aspects of electrical systems, and the IEEE Computer Society is a major subgroup dedicated to all areas of computing: hardware, software, standards, and people.

The IEEE is a major publisher of peer-reviewed research, and produces a large number of technical journals. It is also a major influence on standards and protocols used throughout the computing world, most famously producing the IEEE floating-point format and all of the major standards for network communications. The society takes a leading role in advanced education and accreditation of academic programs in electrical, electronic, and computer engineering.

The society also promulgates a code of ethics, but, since computing is an almost entirely unregulated profession, membership in professional organizations is totally voluntary, and codes of ethics are almost entirely unenforceable. The lack of regulation is proving to be a major source of danger and financial loss in this computer-dependent world, and, although there will be strong objections from established practitioners, it seems inevitable that professional regulation will eventually become mandatory, as it is in all other engineering professions.

Business value proposition

The IEEE provides a wide range of services for IT professionals, academics, and other practitioners. These include the high-quality IEEE publications, which range from the very theoretical peer-reviewed *IEEE Transactions* series to the more popular IEEE magazine format (e.g., *IEEE Computer*), all of which provide leading peer-reviewed articles. The IEEE also hosts conferences and seminars on a wide range of computer-related topics, and has a long history of involvement in the development of standards and protocols. The society has a membership progression from Associate, Member, Senior Member, to Fellow. The grade of Member requires that the person has "satisfied IEEE-specified educational requirements and/or has demonstrated professional competence in IEEE-designated fields of interest," whereas Senior Members need to have demonstrated at least ten years of "professional maturity." The career-development model is a useful mechanism for IT organizations to employ to ensure that their workforce is continuing to develop professionally and being involved in the development of a professional body.

Summary of positive issues

The IEEE provides the IT community with a wide range of resources and a career-development path for its membership. The IEEE is a recognized body for the creation of technology standards and protocols.

Summary of potentially negative issues

Membership by IT workers is voluntary and adherence to its code of ethics is therefore also voluntary.

References

- www.ieee.org.
- IEEE Corporate Office, 3 Park Avenue, 17th Floor, New York, NY 10016, USA.

Associated terminology: ACM, BCS, W3C.

Index

Foundation concept: Database.

Definition: An organization imposed upon data that makes future searches faster or more efficient.

Overview

Data is of no use unless it can be found. If a major city's telephone directory imposed no useful organization upon the data it contained, perhaps listing subscribers in the order in which they originally signed up for service, it would contain exactly the same data as a conventional telephone directory, but would be entirely useless. The few million entries in a telephone directory pale into insignificance when compared with the number of records in a major commercial database. Any significant collection of data relies upon an effective indexing scheme for its usefulness.

Indexing data may involve rearranging that data; the familiar example of the telephone directory illustrates this. The data may be physically organized in alphabetical order (or some other logical arrangement), and this certainly allows effective fast access. However, maintaining a special ordering on a dynamic data set can be prohibitively expensive in terms of time and effort. If the telephone directory had to be recomposed and reprinted every time a new subscriber joined the system, the result would be extremely wasteful.

Most practical indexing schemes involve leaving the original data exactly where it is in a totally arbitrary order, and creating a new *Index file* that contains only the parts of the data necessary for rapid access. For example, if a very large encyclopædia allocated exactly one page per entry, but those pages were totally unordered, a very effective index could be created by simply listing the head-words and corresponding page numbers. The index itself would be in alphabetical order, but the pages it refers to would not. The result would work just as well, but would require significantly less effort to maintain. Ordering the index instead of the actual data makes it possible for the same data to appear to be sorted in two incompatible orders concurrently; for example, a telephone directory may be indexed both on name and on number, so that reverse lookups (look up a number to find the subscriber's name) are just as fast.

When creating a database, the designer must be aware of the kinds of queries and other accesses that are likely to be performed. With this information, an informed decision can be made on what indexes should be created. Any complete DBMS (database management system) will automatically create whatever indexes are requested. Usually, indexes are set up when that database's tables are defined; creating an index after the data has been added can be a computationally intense process. The method described above, namely that of creating an alphabetical list of head-words and page numbers, is very popular in database implementations, and is often known as *ISAM* (indexed sequential access method). Other popular forms of database index are the *B-Tree* (which provides much faster access, but is much more demanding in terms of storage requirements), and the *Hash table* (which is even faster, but does not support alphabetical browsing).

Business value proposition

Indexing is an essential part of data organization, and it takes some technical training in database management to make the right decisions about which indexes should be created. Every index requires significant storage space and adds to the time required to perform an insertion or deletion, so having too many indexes can be a costly mistake, but having too few indexes can result in a query taking literally millions of times longer than it needs to.

While the DBMS software performs the actual tasks of indexing, it is the role of the database administrator (DBA) to make a determination on the trade-offs between system performance and resource requirements. The DBA may then create and

implement an appropriate data structure in an attempt to meet the requirements.

DBMS software is not the only common application of indexing. Commercial web-search engines that perform keyword searches on billions of web pages in a fraction of a second could not do so if the searches were actually performed when requested. Instead, an index is created by pre-searching for all possible words, and saving the results, so that each simple user query may be answered immediately by simply retrieving already-known answers. Many operating systems for personal computers provide an indexing service that, once it has been set up, allows rapid searching for any file containing a particular word or phrase.

Summary of positive issues

Indexing is a well-known method for structuring and organizing data with a large theoretical, academic, and practitioner literature. DBMS systems contain tried and tested indexing algorithms that are capable of automatically indexing data very efficiently.

Summary of potentially negative issues

Poorly designed indexes can lead to a lower overall performance in some circumstances than other methods for file structuring. Index file structures in some implementations may require more direct-access memory space to store their indexes.

References

- C. Date (2000). *An Introduction to Database Systems* (New York, Addison-Wesley).
- R. Elmasri and S. Navathe (1994). *Fundamentals of Database Systems* (New York, Addison-Wesley).
- T. Cormen, C. Leiserson, R. Rivest, and C. Stein (2001). *Introduction to Algorithms* (Cambridge, MA, MIT Press).

Associated terminology: Database, Database administrator.

Information lifecycle management

Foundation concept: CIO.
Definition: Information lifecycle management is the process of managing data from its point of creation to when it is purposely and permanently deleted.

Overview

The ILM concept covers the storage and management of data during its entire lifecycle. Corporate data takes several forms, including live transaction data that resides, for example, on an ERP system; data that has been through an extraction, transformation, and loading (ETL) process and resides in a data warehouse; data that is in backup mode to enable transactional or data warehouse data to be replaced; and data that is archived or held in long-term storage (10+ years) to satisfy legal compliance requirements or other needs.

ILM stresses not only the security of the data but also the managerial issues that surround the lifecycle, assessing the value of the data (its age, whether the data is in live usage, whether the data is warehoused, required for compliance, or merely backed up for completeness), and using the appropriate technology to store the data at minimal cost over its entire lifecycle. This duality of view is necessary because corporate data ownership continues to grow, as do the requirements to keep data for longer.

In order to reduce overheads, managerial and technical, CIOs deploy a variety of technologies. These include *Storage resource management* (SRM) systems that manage the data across the organization and *Hierarchical storage management* (HSM) tools that manage the data upon different types of storage device and facilitate the archiving of data (e.g., from a high-performance disk storage unit to a tape library). As data archiving has grown in importance with the changes in regulatory compliance requirements, the value and urgency of moving

data from one type of storage medium to another has changed. Accordingly, HSM has evolved into what are termed *Automated data migration* (ADM) tools that intelligently move data from one type of data storage to another. These tools are often built upon a *Storage area network* (SAN), a high-performance independent sub-network of a larger corporate network that specifically connects dedicated storage devices together, allowing high volumes of data to be stored, accessed, and moved around a network as required, without impacting the existing corporate network.

Business value proposition

ILM aims to provide complete control and management of the data lifecycle. The use of an ILM philosophy that is backed with SRM and ADM tools allows organizations to achieve several goals. Firstly, they create a system that balances data accessibility with value and need. Secondly, the ILM philosophy helps ensure regulatory compliance with regard to data storage. Thirdly, the systems allow efficient data storage across each data storage level. Fourthly, the systems allow much of the overhead to be automated, allowing the systems manager to focus upon other issues. Fifthly, the ILM philosophy and systems may be designed to map onto an organization's disaster-recovery and contingency plans.

ILM is supported by a variety of tools and vendors, including *Storage service providers* (SSP), which act as providers of storage services, including offering a wide range of services for outsourcing of data storage. These include storing data, providing backup services, and remote data-continuity services via SANs, VPNs, or internet connections.

Summary of positive issues

ILM enables a data-value approach to data storage to be taken. ILM assists organizations with regulatory data compliance. Many vendors support the ILM philosophy through the provision of tools and services.

Summary of potentially negative issues

The development of an ILM initiative can be resource intensive, potentially requiring a company to re-engineer its systems to ensure alignment between the IT strategy and the ILM strategy. The risks associated with corporate data security, location, and ownership over the entire lifecycle need to be assessed.

Reference

- R. Maier, T. Hädrich and R. Peinl (2005). *Enterprise Knowledge Infrastructures* (Berlin, Springer-Verlag).

Associated terminology: ERP, Data warehouse, ETL.

Information Technology

Foundation concepts: CIO, MIS.
Definition: A collection of best practices for the provision of IT service management (ITSM).

Overview

The Information Technology Infrastructure Library[†] is a collection of IT service management best practices that aim to provide the leaders of IT organizations with a top-down business-driven approach to the management of IT processes, people, and technology. The library was originally developed in the 1980s by the UK government's Central Computer and Telecommunications Agency (CCTA). Since then it has been refined and extended by the UK's Office of Government Commerce (OGC), which subsequently published a Version 2 of the ITIL (1998–2004) and has commenced work on Version 3. A British Standard (BS 15000-1:2002) based upon ITIL has also been developed; this standard has subsequently been

[†] ITIL is a registered trade mark of OGC – The Office of Government Commerce.

developed into an international standard: ISO 20000.

The ITIL model is broken down into eight areas:

- service support,
- service delivery,
- planning to implement service management,
- security management,
- ICT (information and communications technology) infrastructure management,
- application management,
- software asset management, and
- business perspective.

Each of the eight components focuses upon a specific aspect of IT operations and breaks that aspect down into services. For each service, best practices are provided at three levels: the strategic, the tactical, and the operational.

The methodology is hierarchically based and allows the selective implementation of only those aspects relevant to the organization. For example, within the service-support component there are subcomponents including help-desk management, incident management, problem management, configuration management, change management, and release management, some or all of which may be adopted and implemented by the IT organization.

While the ITIL does not specify the actual steps to be performed in order to execute the best practice, the operational guidance is sufficient that IT professionals can be confident in their implementations. In the case of help-desk management, the incoming calls from computer users can be divided into appropriate categories, such as laptop, desktop, network, and mainframe, and routed to specialists in each of these areas. The service-level requirements for each area can then be defined (e.g., to resolve each problem in 30 minutes). Should the problem not be resolved, the processes associated with more advanced and continued support are also defined. These may include routing the problem to more advanced technical-support personnel, notifying the appropriate levels of IT and business-process owner of the problem, and identifying alternatives should the problem not be resolved within an acceptable predefined time limit, so that service can be restored to the agreed levels.

The components are managed in conjunction with a database, known as the *Configuration management database* (CMDB), that maintains a "map" of the IT infrastructure within the organization and a log of the problems or "incidents" associated with aspects of the infrastructure. These incidents can then be analyzed and problem areas identified and resolved, further improving the quality of the IT service.

Business value proposition

The ITIL has been used in the UK and the European Union since its introduction in the 1980s and has continued to grow in popularity in other parts of the world. Acceptance and use of ITIL in the United States has been driven by the need to have stronger IT governance in order to accommodate regulatory compliance legislation such as Sarbanes–Oxley and HIPAA.

The ITIL allows organizations to standardize the processes associated with their use of IT. Central to the approach is the provision of standardized definitions for many of the terms used in the IT organization, which allows all members of the organization to have a common basis for discussions pertaining to people, processes, and the IT systems. This common basis of understanding enables all parties to have a more accurate understanding of the status and role of technology within the organization, from which more developed business cases and solutions can in turn be created.

By standardizing on best processes and easing communications between technologists and the users in the business units,

ITIL is aimed at reducing the costs of IT deployment and maintenance while raising service levels. The standards-based approach allows consistent guidance to be provided by the IT organization, and, through certification-based training, it creates a consistent knowledge level within the workforce, also leveraging the ability to recruit certified individuals to support ITIL-based procedures. The standards-based approach to IT service delivery can also be applied to contract vendors delivering outsourced services, in that a contactor can be required to be ITIL-certified. This provides a higher basis for IT service quality and for competitive bidding on service provision due to the processes being fully understood by multiple vendors.

ITIL certification is sanctioned by the ITIL Certification Management Board (ICMB) and is composed of the OGC, the ITSMF (IT Service Management Forum) and the two examination institutes EXIN and ISEB. The ICM has created three levels of certification: foundation, practitioner, and manager. Foundation certification recognizes that the holder is familiar with best practices in ITSM, practitioner certification recognizes that the holder understands the theory of ITSM and how to apply that theory in practice, and the manager certification recognizes that the holder can manage ITIL solutions across a range of service-management areas.

The ITIL is frequently used in combination with ISO/IEC 17999 and the COSO/CobiT frameworks. Typically, ITIL is used as the backbone framework for delivery and support processes in conjunction with ISO 17999, around which security controls are created, while COSO/CobiT is used on controls and metrics that pertain to financial systems and Sarbanes–Oxley compliance.

Summary of positive issues

ITIL is a widely used methodology for implementing best practices in IT service management. There is a substantial literature on ITIL and a growing body of consultants and practitioners to support the implementation of ITIL. Certification of ITIL practitioners is available, this being sanctioned by professional IT bodies such as the British Computer Society and the UK Government's Office of Government Commerce. The ITIL approach to service management is scalable and structured, and allows a standardized and more efficient and effective implementation of IT practices. It can be combined with other IT governance frameworks, including CobiT and ISO 17999.

Summary of potentially negative issues

ITIL provides guidance on how to achieve best practice IT service management, but is not specific on the fine details of implementation. The implementation process can be slow and resource-intensive, and the associated process changes can be disruptive in the short term.

References

- http://www.bs15000.org.uk/.
- http://www.itsmf.com/.
- http://www.ogc.gov.uk/.
- B. Worthen (2005). "ITIL power," in *CIO Magazine*, September 1, 2005.

Associated terminology: ISO/IEC 17999, British Computer Society, Sarbanes–Oxley.

Instant messaging (IM)

Foundation concepts: Email, Client–server.
Definition: An interactive internet communications method, akin to a typed telephone call.

Overview

Whereas email is strictly non-interactive, a message is typed and sent, and the recipient opens and reads it when they feel like it, *Instant messaging* (IM) makes the basic concept of email interactive. Communicating by email is the internet version of

sending telegrams or letters; communication by IM is the internet equivalent of a typed telephone call. Both (or all) participants in communication must be actively using the IM system at the same time; as one types or inserts graphical elements, what they type is immediately shown to the other(s). All participants may, and often do, type at the same time, producing a very confused "conversation."

Instant messaging systems work in one of three modes. In *Character mode*, every keypress is sent to every participant as soon as it occurs; in *Line mode*, nothing is sent until a whole line has been typed, and the enter key has been pressed, giving typers the opportunity to correct mistakes before they are seen; and in *Message mode*, participants compose whole sentences or paragraphs or more, and it is all sent at once when they click a "send" button or key. Message mode allows users to see what they are writing and perhaps think better of it before sending, so it always results in more rational and productive conversations, makes more efficient use of network bandwidth, and is easier to implement correctly. Character mode is generally the most popular because there are no frustrating delays while someone is typing, and it gives the illusion of a real face-to-face communication, although it is exceptionally inefficient. Line mode seems to be the most common choice for IM service providers.

Although instant messaging has become the poster child of the *Peer-to-peer* movement in network communications, many commercial systems in fact use a *Client--server* model: when two participants are communicating, what they type does not go directly from one computer to the other; instead it goes first to a central server, and thence to the intended recipient. This makes implementation easier, and allows the service provider to maintain complete control over all aspects of its use. It also means that communications might not be as private as users would imagine; even if the provider is not spying on communications, the vicissitudes of operating systems can result in messages being accidentally preserved on servers, and these are subject to discovery in legal proceedings.

Just as the popularity of email has been a boon to e-commerce at the same time as proving a great destroyer of office productivity, IM is also a double-edged sword, but this time the positive aspects can be hard to detect. Unlike email, IM requires the full attention of both parties for the duration of the conversation, the same as with telephone calls, except that IM takes far more time than a telephone call because most users are not very good typists. The extreme ease and absence of direct financial cost associated with IM means that people are far more likely to initiate communications for simple matters that could have been resolved with just a few seconds' thought. The immediacy of IM makes serious thought most unlikely.

This does have one possibly positive aspect: because IM frequently involves very little thought, it is sometimes possible to substitute an artificial intelligence (AI) application for a real human correspondent. The famous Eliza experiments conducted by J. Weizenbaum starting in 1965 were simple (by modern standards) programs that simulated real people in conversation. They work by picking out what seem to be key words in what a person types, and reinserting them, transformed, into standard responses selected from a predefined list. The original Eliza simulated a psychotherapist, and was astonishingly successful, eliciting embarrassingly frank admissions from people who should have known better. Modern Elizas can often answer the majority of customer-service enquiries through the same techniques. Organizations must take great care if considering this option, since an Eliza-like system, or any other AI technique yet discovered, will be unable to answer any complex questions that could not have been

answered by simply reading the manual, and may completely alienate customers.

Business value proposition

Instant messaging has become a ubiquitous aspect of communications in today's society. It is easy to use and accessible wherever there is an internet connection. It has taken its place alongside the conventional telephone call and provides a convenient mechanism for communicating when the telephone might not be appropriate, e.g., when a person is at work and a long "chat" to a friend on the telephone could be noticed and contravenes corporate policy. The IM systems can, however, be disabled by network managers and this would be likely to raise corporate productivity. It is unlikely that IM would be advantageous inside a corporation when a telephone conversation is more appropriate. However, when there is a need to communicate over long distances and where the telephone charges are high, IM can provide an advantageous alternative. However, *Voice over IP* (internet telephony) can offer the communicating parties the same advantages as IM but with true voice communication.

In some industries, e.g., the brokerage business, the use of any communication tools requires that the communication be recorded. This prevents disputes over a potential "buy" or "sell" order between a broker and a client. Instant messaging is not immune to this requirement; however, technically this is a more challenging task and requires that procedures to capture and store the interactions be put in place. An organization with moderate IT capability should be able to produce its own proprietary IM system that conforms to any requirements and restrictions they may have.

Summary of positive issues

Instant messaging systems are easy to install on a computer and require only an internet connection in order for them to operate. Instant messaging provides a low-cost communication option.

Summary of potentially negative issues

Abuse of IM by workers can be detrimental to corporate performance. The messages may need to be stored for regulatory compliance. Instant messaging systems provide yet another unsecured channel into a computer system; although it is currently not a real problem, it is quite likely that this will prove to be an exploitable vulnerability in the future.

References

- R. Khan and B. Blair (2004). *Information Nation: Seven Keys to Information Management Compliance* (New York, Aim International).
- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).
- J. Weizenbam (1966). "ELIZA," *Communications of* the A.C.M., January.

Associated terminology: Email, Voice over IP, Peer to peer, Client–server.

Internet

Foundation concept: Network.
Definitions:

Internet: the entire worldwide community of interconnected computer networks.
Internet communications: communications between two systems that are not part of the same local-area network.
Private internet: a group of local-area networks sharing a private IP-address space, and unable to communicate directly with the internet as a whole.

Overviews

Private internet: one of the ways to overcome the global shortage of IP addresses is for an organization to allocate private IP addresses to its own systems. Private IP addresses are those beginning with "10.", "172.16." to "172.31.", and "192.168.". Private IP addresses are valid for use on an

internal network if it is configured correctly, but are not valid for use on the whole internet. Computers arranged in this way are unable to communicate with the outside world except through *Proxy servers* (see *Network Address Translation* and *Proxy* for details), and are said to form a *Private internet*.

Internet communications: communications within one *Local-area network* (LAN) are referred to as intranet communications. Communications between systems that are not on the same LAN are referred to as internet communications. Internet communications can not be based on aspects of LAN protocols, such as ethernet hardware addresses (MAC addresses), and must use higher-level protocols (such as IP, the internet protocol) instead. This merely is a technical distinction.

The internet: The internet is a cover-all term for all of the computers, both servers and clients, and other network-enabled devices, together with the connections between them, that form the global digital communications network.

The internet is tied together by IP, the *Internet protocol*, a system that defines how individual devices are addressed, and how transmissions are routed throughout the network from one device to another. It provides a uniform format for transmissions that is capable of carrying any form of digital data over any kind of connection, with reasonable efficiency.

Every device must have an *IP address* allocated to it before it can participate in the internet. IP addresses may be static (permanently allocated to a particular device), or dynamic (allocated on demand and released after a short period). Blocks of IP addresses are allocated to organizations by an international organization known as ICANN (Internet Corporation for Assigned Names and Numbers), and those organizations then allocate individual addresses within their blocks as they see fit. Individual IP addresses can be allocated directly by ICANN, but that is not usually a cost-effective solution.

Credit for inventing the internet has been awarded to many people. Three different groups, Paul Baran of the RAND corporation, Leonard Kleinrock of MIT, and Donald Davies and Roger Scantlebury of NPL (National Physical Laboratory, UK) seem to have developed the idea of *Packet-switched networks* (see *Packet switching*) independently and at about the same time, around 1962. In 1968, the Advanced Research Projects Agency (ARPA) of the US government funded the first internet-like computer network, which was known as ARPANET, and came on line in 1969, connecting UCLA (University of California, Los Angeles) and SRI (Stanford Research Institute), soon joined by UCSB (University of California, Santa Barbara) and the University of Utah. ARPANET gradually expanded over the years, linking with other national networks, until it grew to be what is today called the internet.

The TCP/IP protocol suite was invented by Vinton Cerf of Stanford and Bob Kahn of ARPA (by then renamed DARPA, the D is for Defense) starting in 1973, and in 1974 the word "internet" was used for the first time in their published paper describing TCP. In 1983, the University of Wisconsin introduced the *Domain-name system*, which completed the basic infrastructure of the modern internet. The system that is considered by many to give life to the modern internet, the "World Wide Web," was not invented until 1990 (see *World Wide Web* for details).

The *Backbone* of the internet consists of a large number of major switching centers each connected to a moderate number of others. Many smaller switching centers may be connected to each of the major ones, and many smaller yet to each of them, down to the level of the LAN. Each individual computer generally has one communications path available to one of the major switching centers, but the major

switching centers are very highly interconnected, and may communicate with one another over a wide variety of paths, choosing whichever is best at any given moment. This is what gives the internet its strength: any switching center could fail, and only those individual computers relying on it would become disconnected; the internet as a whole would be undamaged, since there will always be a variety of paths that avoid any out-of-service node. One of the original purposes behind packet-switched networks was to provide a military command structure that would still be operational after a disastrous nuclear attack.

Since the World Wide Web converted the internet from a governmental and academic research tool into a wildly popular public utility, and email became a mass communications (and miscommunication) medium, the original purpose of the internet has become compromised. This has led to the creation of the *Internet2* consortium, devoted to the creation of an advanced internet, with the intent of fostering new technologies in both hardware and software that will improve the speed, reliability, safety, and bandwidth of the internet.

As the original internet became deregulated, the US military decided to create its own secure "internet" known as SIPERNET (Secure IP Routing Network). It is intended to link all five branches of the military in one network and possess greater connectivity for mobile forces, connecting ship to shore, and field troops to operational bases and to headquarters through satellite and other mechanisms. A second military internet was created for the Navy and Marines and is known as NIPERNET (Navy Internet Protocol Router Network). A third military internet is the JWICS network (Joint Worldwide Intelligence Communications System); through it the US military securely transmits data classified as top-secret to specific designated recipients. Naturally, much of the technology, architecture, and cost of these systems is classified.

Business value proposition

The internet has enabled individuals, corporations, and entities within corporations to connect together in many new and productive ways. These include *Online communities* in which virtual communities unite members to solve problems, discuss issues, and foster commerce. The internet has spawned a whole new mechanism for the development and delivery of business models and commerce including *Collaborative commerce*, *Business-to-business* (B2B) commerce, and *business-to-consumer* (B2C) commerce.

The technology has grown to include new mechanisms for connecting to the internet and protocols such as *WAP* allow mobile devices using *Micro-browsers* to receive and send information over wireless internet connections. The technologies that run the internet such as *TCP/IP* have allowed companies to base their intranet and internet architectures upon one consistent set of protocols, and this relieves them of the burdens associated with managing multiple protocols and their interactions. Languages such as XML have revolutionized the way data is transmitted, releasing organizations from outdated fixed formats or proprietary data formats.

Summary of positive issues

The internet provides an open-standard-based architecture for inter-computer communication. The internet is available universally via landlines or wireless devices (including satellite phones). The internet protocols have simplified network management for CIOs and individuals.

Summary of potentially negative issues

The internet has morphed into a commercial, academic, and public-domain space that is becoming more congested, and its users are often subject to those with

malicious intent, e.g., viruses, worms, Trojan horses, phishing, and the use of network sniffers for those wishing to attempt internet fraud.

References

- K. Hafner (1998). *Where Wizards Stay Up Late: The Origins of the Internet* (New York, Simon and Schuster).
- D. Groth (2003). *A+ Complete* (Hoboken, NJ, Sybex–John Wiley and Sons).
- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).

Associated terminology: Virtual private networks, World Wide Web, Internet protocol.

Internet protocol (IP)

Foundation concepts: Internet, Network, Protocol.
Definition: The internet is the worldwide community of computers interconnected by telephone lines, cable, satellite, etc. The internet protocol is the set of established rules and procedures that makes the internet work, by establishing a universal communications language and addressing scheme

Overview

The internet exists as a large heterogeneous collection of interacting technologies, and is likely to continue to do so for the foreseeable future. When computers are physically close together, a simple, cheap copper wire connection between them gives excellent results. When computers are widely separated and their owners do not have large budgets (home users particularly), a connection through the public utilities (e.g., telephone lines) is often the only viable solution. When major businesses are spread over geographically large areas, no physical connection is viable, so wireless (satellite) connections are the method of choice. All of these different physical connection media work in different ways, and it would not be at all practical for every computer on the internet to "know" exactly how to communicate with all the others.

To make communications practical, a *layered* system has been developed, and the *Internet protocol* (IP) is one of its fundamental components.

At the lowest level, every computer has the ability to communicate along the kind of connection that it has, and no other. Typically, a small-to-medium office will have a group of computers all connected together on a *Local area network* (LAN) that consists of cheap copper wires and small electronic components called *Switches* and *Hubs*. This collection of wires and hubs connects together only the local group: it does not extend to any great distance, and the major arteries of the internet certainly use much more sophisticated technology. Since all the computers in the local group have the same kind of connection, they could easily communicate with each other, but not with anything outside the group.

Every local group of computers has one special member, sometimes a normal computer with two network connections, sometimes a special-purpose piece of hardware. This device with two network connections forms a *Bridge* or *Gateway* between one LAN and another, or between a LAN and a larger internet artery. This one device must, of course, be capable of communicating on both of the networks it is connected to, and they may be using completely different technologies. The purpose of the IP is to provide a single uniform "language" that allows all of the other computers to make use of this gateway. A computer on one side of the gateway might be capable only of communicating along cheap copper wires; the computers on the other side may (for example) be capable of communicating by a high-bandwidth satellite link, perhaps even using technology that didn't exist when the other computers were made. When a "copper wire" computer needs to communicate with a "satellite" computer, they can not use their own built-in methods, because

they are totally incompatible. Instead, they use the IP.

The IP strictly defines message formats that are independent of the technology being used. A copper-wire-connected computer composes a message in the IP format, then sends it to the gateway of its own LAN over the copper wires that connect it. The gateway computer receives the message, and, because the IP is universal and invariant, it can understand the message, and see that it needs to be sent to one of the computers on the satellite side. It is capable of communicating both along copper wire and by satellite, so it simply "re-wraps" the IP message in the specialized satellite message format, and sends it on its way.

Typically, a message transmitted over the internet will make a few dozen such *Hop*s through gateways between LANs and internet arteries before it reaches its destination. The IP completely defines the universal message format that allows this to happen, and also provides a universal addressing scheme.

It is essential that computers on the internet should have fixed, simple addresses that uniquely identify them, just as homes and businesses must have known addresses to receive their mail.

The internet protocol uses simple large numbers as addresses. Usually, these numeric addresses are seen as four smallish numbers separated by dots (e.g., 127.35.101.98), but this is just a notational convenience. It is really just one big number split into four parts to make it easier (for humans) to handle without error. Every computer on the internet at any time has one of these numbers assigned to it and no other. The IP tells the *Gateway* computers how to correctly forward any message; given the IP address of the destination, it is easy for any gateway to work out which one of its neighbors should receive the message in order for it to eventually reach its intended recipient.

There is some logic to address assignment. A large company with many thousands of computers may be given all the IP addresses that begin with (for example) 127.35 (known as a *Class B* address) and allowed to allocate them as they see fit. A smaller company may be given a *Class C* address, which would give them control over all the addresses beginning with (for example) 127.35.101. An individual computer owner may be given a single full address for their own personal use.

The current IP addressing scheme combines four small numbers (as in the example above) to make one numeric address. Because each of the small numbers has a restricted range, the total number of possible IP addresses is something under 4 000 000 000. That seemed an absurdly large number at the time, but is much less than the current population of the world. If the average number of computers per person exceeds about $\frac{2}{3}$, there just won't be enough addresses to go round. Since many things that are not really computers are now internet-connected, the world has already nearly run out of IP addresses. The next generation of addresses, combining 16 small numbers to make one numeric address, is known as IP6, and is already being implemented. This will provide an enormous number of different addresses (over 300 000 000 000 000 000 000 000 000 000 000 000 000).

ATM (*Asynchronous transfer mode*) is another routing system, which was originally intended as an alternative or even replacement for IP. Under ATM, data is divided up into much smaller packets, all with a fixed size (48 bytes of data plus 5 bytes of header information), called *Cells*. Before data is communicated between two sites, a *Virtual circuit* is set up, which specifies the communications path to be taken; then all data uses that virtual circuit, following the same path from source to destination. These two changes allow greater control over the overall transmission rate, and this

can significantly reduce *Jitter* (signals breaking up and becoming "choppy") in audio and video signals. However, improvements in WAN bandwidth made IP work much more smoothly, and ATM is a very complex protocol, so it did not achieve its proponents' goals, and IP still reigns.

Business value proposition

Access to the internet, through the IP, is almost a *sine qua non* for any modern business. The use of internet protocols allows a clear and simple technology strategy to be adopted by the whole organization. The acknowledgement of IP as the standard protocol for messaging within and between organizations allows the internal IT organization to focus its protocol monitoring on just one set of technology protocols. The IP will continue to evolve and advance in the future, and a corporate IT group needs to monitor these changes. IP working groups have been set up by major software and technology vendors such as The Internet Engineering Task Force to discuss and mold the shape that the future IPs will take. The IP is fundamental to all devices and software that operate over the internet; thus all vendors will need to comply with the standard as it evolves, and there are no seriously competing protocols in this area. In the future, a major area of concern for business users will be in the transition from IP4 to IP6, ensuring that devices connect correctly and that all of the software is upgraded.

Summary of positive issues

The IP is what makes the internet work. All of the positive and negative issues for the internet itself are essentially issues for the IP.

Specific to the protocol is the fact that it is very simple and universally established. This means that any two computers that can be connected together can be expected to be able to communicate in a meaningful way, without any specialist or non-standard tools or software.

Summary of potentially negative issues

The IP was designed at a time when nobody anticipated the vast number of computers that would eventually be interconnected. As a result, its universal addressing scheme is rapidly running out of addresses to use. IP4 (the current version) will have to be replaced by the already designed and implemented IP6, or some other alternative. Although both IP4 and IP6 are well known and stable, the potential for disastrous upheaval as one system is replaced by the other is as great as the potential for disaster was with the Y2K bug. It will probably turn out to be a problem-free transition (like Y2K turned out to be), but nothing is guaranteed until it is all over.

References

- http://www.ietf.org/.
- W. R. Stevens (1994). *TCP/IP Illustrated* (New York, Addison-Wesley).
- D. Groth (2003). *A+ Complete* (Hoboken, NJ, Sybex–John Wiley and Sons).
- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).

Associated terminology: Internet, TCP, DHCP, Network.

ISO/IEC 17799

Foundation concept: Security.

Definition: ISO/IEC 17799 is an international standard intended to provide guidance for IT professionals in establishing a set of security processes and policies for their organization.

Overview

The ISO/IEC standard 17799 has its origins in the British Standard BS7799 originally developed by the UK's Department of Trade and Industry. The standard is really a code of practice that aims to provide

a comprehensive set of guidelines that IT organizations can follow in order to provide a high degree of information security.

The code of practice itself is comprised of ten sub-sections that cover

(1) Security policy,
(2) System access control,
(3) Asset classification and control,
(4) Personnel security management,
(5) Organizational security,
(6) Computer and operations management,
(7) Physical and environmental security,
(8) Systems development and maintenance,
(9) Business continuity management, and
(10) Compliance management.

The issues surrounding *security policy* are centered upon providing information and data to the organization relating to information security that can be used to create an overall security policy. The creation of the post of chief security officer may follow the initial assessment of a corporation's information security performance level.

The ability to control *systems access* is vital to corporations wishing to be compliant with Sarbanes–Oxley and other US regulations, which require organizations to document and enforce strict access control.

The third factor, *asset classification and control*, requires organizations to identify all corporate information assets, and define security levels for those assets. The code also presents guidelines pertaining to *personnel policies* and security that include issues such as performing background checks on employees and enforcing non-disclosure agreements, as well as documenting and investigating systems errors.

The code of practice details a set of guidelines relating to the *organizational security* and access to information, including such issues as the establishment of management policy groups, the delegation of information security ownership, and the policies required to enable third-party access to corporate information systems.

The development of policies pertaining to the *operational* aspects of the information systems within a corporation is central to the code of practice and covers an extensive array of issues, including capacity management, provision of security steps to prevent malicious attacks upon the network, establishment of incident logs, and the establishment of physical and system-based security measures to protect intellectual property.

The establishment of practices to protect the physical aspects of the information systems is also important, and the code discusses areas of concern that need to be addressed, including the protection of systems using high-security areas, network technologies, physical protection mechanisms, and enforcement of equipment use policies.

The guidelines also pertain to *systems development and maintenance*, and stress the need to build into code high degrees of security as well as mechanisms for data validation and verification. The code also covers contingency planning and the mechanisms behind compliance. The adherence to these guidelines can be certified by organizations such as the British Standards Institute (BSI).

The ISO/IEC code of practice is a very high level set of guidelines and requires extensive "filling in" through other technical standards and policies such as those proposed and developed by the US National Institute of Standards and Technology (NIST) and published in the Special Publication 800 series. While no code of practice, systems policies, systems, or people for that matter can be considered to have a zero security risk, the guidelines when combined with practical steps can provide the first steps toward developing a secure IT organization.

Business value proposition

The need for information security across the organization is of paramount importance to all organizations and the ISO/IEC 17799 code of practice provides a framework covering all aspects of information security, including the physical issues, software issues, human-resources issues, and the establishment of internal policies and processes. The code of practice describes the macro issues that need to be addressed individually with physical and practical processes such that a complete security policy is enacted and enforced.

Summary of positive issues

The ISO/IEC 17799 provides a code of conduct that helps organizations develop a set of security policies and procedures. Companies that adhere to the code can be audited and certified by independent entities. The code is comprehensive, covering both physical and intangible aspects of security. The code can be further developed and supported by combining it with more technical standards and security process models.

Summary of potentially negative issues

The standard is very high level and does not detail specific technologies. The standards are not legally mandated and adherence is voluntary.

References

- International Standard ISO/IEC 17799:2005 Code of Practice for Information Security Management, www.iso.org.
- US Department of Commerce (2005). *An Introduction to Computer Security*, NIST SP 800–12 (Gaithersburg, MD, US Department of Commerce).

ISP (Internet service provider)

Foundation concepts: Internet, Broadband.
Not to be confused with: Hosting.

Definition: An organization that provides internet access to its customers.

Overview

Internet access is an expensive commodity. The equipment, staffing, technical, and administrative requirements for establishing and maintaining a node on the internet *Backbone* are onerous to say the least. For a large corporation, it makes financial sense to set up an in-house access node, but for others the cost is prohibitive. For individuals, the cost of internet access would be completely unaffordable without some way to share the burden.

An *Internet service provider* (ISP) is simply an organization that has invested in its own internet access node, and sells to others the rights to use it. Of course, some means of connection between the customers' computers and their site must be provided. In some cases customers make a dial-up connection using a modem and the public telephone system. In others some high-bandwidth or broadband infrastructure is used, such as existing cable-television cables, enhanced telephone lines (such as DSL), or specially installed lines (such as ISDN and T1).

An ISP may simply provide internet access and nothing more, but more often it will provide secondary value-added services such as email servers and data backup facilities. Charges may be based on a monthly flat fee, a bandwidth usage fee ("per megabyte," etc.), or a combination of the two. ISPs sometimes also provide *Hosting services*, but that is a different but related line of business.

Business value proposition

ISPs provide a mechanism for companies and individuals to access the internet without having to set up their own internet access node, which is prohibitively expensive for all but the largest companies. The selection of an ISP is an important

consideration for a business. Unfortunately, in some markets there is a limited choice, by regulation, because of limited backbone access due to incumbents with embedded ownership, or simply through the dynamics of supply and demand. When ISP options are available the cheapest company might not always be the best; similarly the most expensive might not offer a truly superior product and thus metrics beyond simply the financial need to be employed. Frequently companies allow users a grace trial period during which they can connect for free. During this period system connectivity tests can be made, such as running a "speed-tester" application to ensure that the speed of the connection is as advertised.

Sometimes ISPs advertise the speed of their systems in asymmetric terms such that downloads are one speed while input from the user is another; if bi-directional speed is important then the true speed of the system connection need to be established. A second issue is the nature of the connection in terms of down-times. Some companies offering high-speed connections such as DSL offer a dial-up backup option when DSL is unavailable. The nature and quality of customer service also needs to be established – if you're having problems with a connection, will the ISP be there 24×7 to assist? The contract with the ISP needs also to be examined to ensure that the ISP adheres to the level of privacy you expect. While legislation exists in the United States, European Union, and many other countries, it is by no means universal and, should corporate secrets or even what could be considered "free speech" be transmitted through VoIP, strong encryption is recommended.

Other issues that can influence the selection of an ISP include the scalability of the ISP's capacity, since a rapidly growing company may rapidly outgrow a small ISP. What additional services are offered by the ISP (e.g., email server, firewalls, security) and how long the ISP has been operational are also reasonable considerations. It may be worthwhile for a company locating in an emergent market to ask for references from the ISP. However, in mature markets the costs of switching from one ISP to another are usually relatively low.

Summary of positive issues

ISPs offer an easy and relatively inexpensive mechanism for connecting to the internet. There are many ISPs in developed economies. Costs of switching from one ISP to another are low. Many ISPs offer many additional services beyond connection services.

Summary of potentially negative issues

Some markets have limited ISP access. In some markets, ISPs are owned and operated by governments that regulate and monitor the traffic that can be passed through them. The speed of the connection offered by some ISPs may fluctuate depending upon load and conditions on the network.

Reference

- T. Casey (2002). *ISP Liability Survival Guide: Strategies for Managing Copyright, Spam, Cache, and Privacy Regulations* (New York, John Wiley and Sons).

Associated terminology: Hosting, Internet, Broadband, Network, Modem.

Java

Foundation concept: Programming language.
Definition: A popular object-oriented programming language, derived from C++ but with many significant differences, some of them improvements.
Not to be confused with: JavaScript.

Overview

For many years, C++ was the only programming language available for large-scale programming projects that properly supported modern programming methodologies, apart from the very unpopular *Ada*. For all its popularity, which continues unabated, C++ has many faults, which can not possibly be rectified without changing it into a fundamentally different kind of language. C++ has many safety features, which would go a long way toward making programs more reliable, but none are enforced; programmers can and do override them at will. C++ carries with it the unfortunate legacy of older languages (most of the old C language survives as part of C++), and it almost encourages an unstructured overcomplicated style of programming. The language itself is exceptionally complex (the official standard fills 774 pages), and very few expert programmers are aware of all of its rules.

Java was developed by a small team at Sun Microsystems, and released to the public in 1995. It was intended to do many good things: remove or replace the worst features of C++, be a relatively simple language that can be completely known and understood, provide safety features that programmers can not choose to ignore, fully embody the object-oriented design methodology, provide automatic cross-platform compatibility for all programs, provide a uniform simplified access to "windowy" graphical user interfaces, be "internet-friendly," provide an extensive thoroughly tested library of programming utilities, and allow secure execution, guaranteeing that programs from untrusted sources can not do any harm. Surprisingly, it succeeded admirably in nearly all of these objectives, with the result that Java is probably the second most popular general-purpose programming language today (some surveys already put it first), and is still gaining ground.

Java programs are not compiled in the traditional sense of being translated into code that can be directly executed on a particular real computer. Instead Java uses a *Virtual machine* (the JVM) designed to provide optimal support for Java-specific features; Java programs are compiled to code executable on the virtual machine, called *Bytecode*, and a JVM emulator application is responsible for executing the code. As a result, Java programs usually run rather more slowly than those written in more traditional languages, but this difference is almost canceled out by the use of JIT (*Just-in-time*) compilers, which convert the bytecode to normal machine code immediately before execution.

The use of a virtual machine to support execution of programs has two major advantages. One is that Java programs will run identically on all computers, requiring only that the JVM has actually been implemented for that platform; this is an advantage that no other general-purpose programming language can boast. The other is that security may be completely controlled. A normal executable program, once it has been allowed to run, has complete control of the computer and can do anything from functioning correctly to erasing the disk drive and sending obscene emails. A virtual machine retains control, and allows the user to control exactly which operations are permissible. Java programs can be run in a totally secure mode in which they are not allowed to access any disk files or make any internet connections, or they can be run in trusted mode, allowing them

to do anything, or they can be run at any intermediate security setting.

The guaranteed cross-platform compatibility and control of security make Java programs very web-friendly. A Java program may be embedded into a web page and automatically run when that page is accessed. This allows much greater functionality than a normal passive web page could provide. The program itself is run through the JVM, so it will work equally well on all platforms, and the user retains complete control of all security settings, so they can happily let the program run, secure in the knowledge that it can't do anything they wouldn't want it to do. Smallish Java programs designed to be accessed from web pages in this way are called *Applets*, meaning "little applications," with the "-let" suffix as in "piglet."

The only serious widely upheld complaints about Java were that its libraries were produced perhaps too quickly, and for many years the standard libraries remained very "buggy" while new features were being cranked out at high speed. The major remaining complaint is that Java very heavy-handedly enforces some extremely arguable policy decisions: it reports as errors things that are common and accepted programming practice, and certainly not wrong; some of the essential libraries make decisions that put speed of execution above flexibility and clean design. These libraries can not in any practical way be replaced by the programmer, and force some very unfortunate implementation practices on experienced expert programmers.

Business value proposition

In 1995 the release of Java coincided with the deregulation of the internet and these two events radically changed the way that organizations thought about their software development. Java and the need for programs to address the internet in essence marked the end of the old mainframe-era languages of Cobol and Fortran. Simultaneously it moved programmers toward a more structured and controllable methodology of program design.

Java also provided an open-source alternative to traditional programming environments associated with C and C++ with which programmers had wrestled for years. One of Java's strengths is that it facilitates the creation of powerful and flexible code for internet-based applications. The strength of Java for CIOs is based upon the portability of the system: the code is compiled down to a form that is capable of being run on nearly any device (including some Java-enabled telephones). This style of development alleviated the platform-specific problems that in the past were so problematic for IT organizations. For many organizations, a lack of cross-platform operability was taken as sufficient reason for staying with an outdated software–hardware platform combination for far too long.

A key to the language's success is the ability of programmers to learn it quickly, especially if they had previous experience with C++. Additionally, programmers like the availability of software libraries which keeps them from having to "reinvent the wheel" and enables them to focus upon more specific and important tasks. This clearly raises productivity and systems reliability, since many of the library functions have been improved over the years as programmers find and document bugs and problems.

Summary of positive issues

Java is free; the compiler, libraries, and virtual machine for a wide variety of platforms may be downloaded from Sun's web site. There are many online resources to help Java programmers, and very many books, including technical references, introductions, and academic texts, that

explore all aspects of the Java programming experience.

Summary of potentially negative issues

The official documentation of the required libraries is not always of the highest quality. For example, the documentation for JDK (Java Development Kit) 1.4.1 and others "explains" the "get preferred size" method of the component class thus: "Gets the preferred size of this component. Returns: a dimension object indicating this component's preferred size." That is not documentation, it is simply repetition. That is not an atypical example; Java programmers must have reliable independent sources of information. The need to scan multiple sites for clues about how to work around bugs in the libraries was for a long time a notable drain on productivity.

The use of Java by some web sites means that a Java virtual machine is required to be installed on client systems. Access to such a machine requires a visit to the Sun Microsystems web site, since some other operating systems vendors who have popular browsers do not support the Java initiative and it can not be downloaded from their sites.

References

- J. Gosling, K. Arnold, and D. Holmes (2005). *Java Programming Language*, 4th edn. (Boston, MA, Addison-Wesley Professional).
- http://java.sun.com/.

Associated terminology: C++, Object-oriented.

JavaScript

Foundation concepts: Dynamic web pages, Programming languages.
Not to be confused with: Java.

Definition: A very basic programming language, based on Java, but much restricted, used to add simple dynamic content to web pages.

Overview

JavaScript is a simple scripting language barely based on Java. It is rarely if ever used for any serious programming tasks; its main use is for small fragments of executable code (scripts) that are to appear inside normal web (HTML) pages, and make things happen under certain circumstances. The actions triggered by JavaScript scripts are usually annoying graphical devices that many viewers would be happier without, such as images being replaced by others at set periods, to display a succession of advertisements in one place, or cursors changing shape, or captions changing in front of the viewer's eyes.

When used with care, JavaScript can add positive value to a web page, and can simplify web development somewhat. It can be used to create tasteful, unobtrusive graphical layouts, and to take proper control of "pop-ups." It is simply the fact that the most basic abilities of JavaScript are so easy to learn that attracts frequent over-use.

JavaScript, being an active element on a web page, is very often a vector for virus or spyware delivery. If not limited, simply viewing a page that contains a script is enough to allow that script to install illicit software on a computer and arrange for it to be effectively unstoppable. Web surfers must take active control of web browser security settings, and ensure that JavaScript is either disabled, or at least forbidden from accessing disk and making its own network connections.

Business value proposition

JavaScript was created to enable programmers with little training or expertise to rapidly create small dynamic web pages. The applications are typically intended to be graphical in nature and

add informational or decorative value to the viewing of a web page. Unfortunately, the very ease with which these systems can be created and their ability to deliver web content via the internet have allowed malevolent users to create scripts containing viruses, spyware, and harmful mistakes. Hence many users prohibit the acceptance or viewing of emails and other incoming data streams that contain JavaScript elements, and that clearly reduces their impact. Network managers and users must be made aware of the risks associated with unrestricted scripts, and a security policy should be enforced to diminish their threat potential.

Summary of positive issues

JavaScript is easy to learn and allows quick development of basic dynamic web pages.

Summary of potentially negative issues

JavaScript can be a very inefficient and ineffective programming language. It should not be considered a general-purpose programming language and is typically useful only for small applications. Hackers have frequently embedded viruses and other malevolent applications within JavaScript elements in web pages. Consequently many computer users have learned to disable JavaScript in order to protect their computers; this has the consequence of making the pages unviewable.

Reference

- D. Flanagan (2001). *JavaScript: The Definitive Guide* (Sebastopol, CA, O'Reilly Press).

Associated terminology: Hypertext.

Joint application design (JAD)

Foundation concept: Software-development lifecycle.

Definition: Joint application design is a collaborative methodology for the specification of software systems.

Overview

Joint application design (JAD) originated with Chuck Morris of IBM Raleigh and Tony Crawford of IBM Toronto in the late 1970s and was developed by IBM Canada in the early 1980s. It uses a collaborative style of system specification development under which the parties involved come together in structured workshops or sessions to determine a system's specification.

Many variants of the original JAD methods have been developed by researchers and consulting organizations, but all share a common focus upon the conceptualization, design, and specification stages of the software development lifecycle. The JAD method breaks the development down into a series of phases.

The first stage is the *Conceptualization* stage (also termed the *Project definition* stage and the *Requirements gathering* phase), in which the project members are identified and then educated on the JAD process. The JAD session then attempts to define the system's boundaries, the basic processes, and high-level data flows. A document is produced that then acts as the requirements document for stage two. Stage two is the *Research* phase in which stakeholders (e.g., function owners, managers, and information-systems personnel familiar with the domain area) are interviewed, with the aim of determining the project's parameters. Having developed a weak specification, the third stage can be undertaken: in this *Preparation* stage, the structure documents and questions to be asked of the team in the fourth stage, the JAD session itself, are created. In the fifth stage the specification document is prepared.

The JAD session itself (stage four) follows a detailed agenda and the workshop participants implement a predefined set of procedures that must be closely followed. The participants include but are not limited to a *Facilitator*, who is knowledgeable in the JAD methodology and acts to drive the discussion and ensure that procedure

is followed in the sessions; *Subject matter experts*, who understand the business function being addressed by the system; *End users*, who ultimately will use the system; *Developers*, software and hardware specialists who will create the system; a *Scribe* who records all decisions and compiles the documentation; and a *Tie breaker*, who is the final arbiter and usually a member of senior management. Additionally, *Observers* are usually present but do not participate in active discussion. The sessions are intended to be peer-to-peer meetings where participants can express themselves freely regardless of rank.

Business value proposition

The JAD methodology has been a popular approach to systems development, especially among consultants. The approach can be used for development of new systems and of new procedures that will link to existing systems, and as a mechanism to perform systems maintenance. The approach is well known and a wide variety of resources is available to support the development approach.

Summary of positive issues

JAD is a well-known and well-supported method for systems development. It focuses on the problem and brings all stakeholders together. JAD sessions help to develop systems that are supported by good documentation.

Summary of potentially negative issues

The JAD approach works well for small applications, but can incur a significant resource overhead for the developer in development of larger applications.

References

- J. August (1991). *Joint Application Design: The Group Session Approach to System Design* (New York, Yourdon Press).
- J. Wood and D. Silver (1989). *Joint Application Design* (New York, John Wiley and Sons).

Knowledge-based systems

Also known as: Expert systems.
Foundation concept: Artificial intelligence.
Definition: Knowledge-based systems are programs that aim to perform at the same level as a human expert over a limited knowledge domain.

Overview

Systems that could perform at levels requiring high degrees of intelligence were originally proposed as an aspect of artificial intelligence (AI) in the 1940s by Alan Turing. As general and steady progress was made toward this aim during the period 1950–1980 the concept of a program that attempts to replicate the level of performance of a human expert over a limited domain, e.g., a system that helps diagnose faults in an automobile engine at the same level as a trained mechanic, became popularly termed an *Expert* or *Knowledge-based* system.

Knowledge-based systems are composed of a knowledge base, an inference engine, a human–computer interface, and, usually, an explanation mechanism. Various complex methodologies have been produced by the research community to assist the *Knowledge engineer* in developing their systems, but the basic process consists of four stages. First, the knowledge engineer elicits knowledge from a domain expert. This process may involve interviews, reviews of manuals, and the examination of case studies. The result of the elicitation process is knowledge in the form of rules, heuristics, facts, etc., which in stage two is refined and encoded in a form known as the *Knowledge representation* that may be manipulated by the inference engine. One popular knowledge representation is the use of simple "if . . . then . . ." rules (e.g., IF light_brightness = dim THEN check_battery, IF light_brightness = dim AND battery = charged THEN check_alternator).

Once the knowledge base has been created, it is necessary in the third stage to build an *Inference mechanism* that is able to manipulate the knowledge relating to a particular problem under consideration in a way that may eventually solve the problem. This typically occurs through a structured dialog between the system and the user, composed of questions such as "Are the lights dim?" The inference engine takes the input and uses the knowledge represented in the knowledge base to arrive at a solution to the problem. The final stage in development is the *Verification* stage, in which the knowledge engineer ensures that the system works correctly. This is a difficult problem because many systems are highly complex and use multiple forms of knowledge representation together with specialized reasoning techniques and may employ fuzzy logic and involve problem areas in which the implementor is not expert.

Early systems were created through the use of AI programming languages such as LISP and Prolog, and during the 1980s special computers known as *LISP Machines* were developed by Symbolics and Xerox to support the development of AI and knowledge-based systems. Although customized systems were developed through programming in LISP and Prolog, this was seen as a slow and difficult process and it was eventually realized that the inference engine and interface components could be standardized in significant ways. This led to the creation of expert-system *Shells*, which incorporate a general purpose inference engine that works for any knowledge base so long as it is encoded in one specific representation. Knowledge engineers then only have to tune the questions that the interface will ask of the user and the responses that it will produce, and the system is complete. This led to somewhat inflexible systems, but it simplified the process considerably and became the

development method of choice for many organizations whether or not they lacked the resources to create customized systems. Successful expert systems are used in a wide range of domains, including the diagnosis of system faults and preparation of schedules for workers, and to assist medical staff.

Business value proposition

Knowledge-based systems are useful programs that can be used to encapsulate corporate knowledge in a form that may be consulted on a variety of corporate problems by any user. The knowledge in a knowledge base acts as a repository that is helpful to companies that have valuable human experts but whose knowledge can not be duplicated in any other way; the human experts may retire, be overloaded, or otherwise indisposed, but their knowledge remains available for consultation. Knowledge-based systems can be used to relieve the human expert of routine tasks and allow them to focus upon truly expert activities.

Summary of positive issues

The literature on knowledge-based systems is well developed, and significant work has been performed on design methodologies that facilitate their specification and creation. Significant research has also been performed on the verification and validation of knowledge-based systems, which is an important aspect of any systems design, especially in the case of systems used in situations with strong constraints and requirements (such as the closedown routines for nuclear power stations). The development of knowledge-based systems has been eased by the availability of programming environments that support the creation of the knowledge base through tool sets, and provide pre-built inference mechanisms that operate on a specific knowledge representation.

Summary of potentially negative issues

Knowledge-based systems rely upon knowledge, and the knowledge elicitation process can be long and difficult. It is important that developers employ rigorous development methodologies and techniques. An expert system's ability to explain its conclusions can be weak, and this justifiably inhibits users from trusting results in critical situations. The knowledge base of a system may, depending upon the domain, need to be updated and maintained, incurring a recurring overhead for the developers.

The maintenance aspects of the lifecycle also need to include a verification phase to ensure that no conflicts or redundancies exist in relation to the specification. A potential problem with knowledge-based systems is that, since they are computer-based, over-reliance on their output without question can be misleading and at times dangerous, especially at the outer edges of the system's knowledge where true expertise is needed. A poorly constructed system may be fragile and might not present the user with an assessment of the validity of the solution proposed. People have a tendency to believe what a computer tells them: a poorly constructed knowledge-based system that provides wrong answers can be much more harmful than a similarly inept human who gives exactly the same wrong answers.

It is essential to keep in mind that genuine expertise is not merely a matter of memorized facts and procedures. A true expert in most domains is able to solve, or at least invent reasonable approaches to, problems of kinds that they have never seen before. This is an aspect of true intelligence, and is outside the scope of any knowledge-based system.

References

- R. Plant and R. Gamble (2003). "Methodologies for the development of knowledge-based systems 1982–2002,"

Knowledge Engineering Review, Volume 18, Issue 1.
- S. Murrell and R. Plant (1997). "A survey of tools for validation and verification 1985–1995," *Decision Support Systems*, Volume 21, No. 4.

Associated terminology: Artificial intelligence, Machine learning, Knowledge engineer, Logic programming.

Knowledge engineer

Foundation concept: Knowledge-based systems.
Definition: A knowledge engineer is an information systems professional who works in the area of artificial intelligence, eliciting knowledge from domain experts and representing that knowledge in a computer system.

Overview

Knowledge engineers perform the task of extracting information from domain experts. This process is known as *Knowledge elicitation*. A variety of elicitation techniques are used, including interviews, document reviews, and structured techniques such as the *Repertory grid* (a method for extracting knowledge in which experts compare the similarities and differences between two sets of task-related parameters within a grid structure). Having elicited the knowledge, the knowledge engineer then performs a careful analysis and structures that information so that it can be represented effectively in a data structure on a computer. These data structures are known as *Knowledge representations* and include *Production rules*, *frames*, and *Semantic networks*. The knowledge engineer is also responsible for maintaining the system and verifying that the system performs correctly.

Business value proposition

Knowledge engineers perform the valuable task of compiling scarce knowledge and placing that knowledge into a computer system. The resultant systems, if well constructed, provide expert levels of performance that can be duplicated at multiple locations and applied to multiple problems. The systems can also act as corporate knowledge repositories.

Summary of positive issues

The knowledge-based systems community has a mature literature and application tool sets have been developed to support the creation of such systems. Faster and more accessible hardware and software systems have facilitated the wider development and use of knowledge-based systems.

Summary of potentially negative issues

The acceptance of knowledge-based techniques remains limited due to the specialized systems requirements and the high cost of development. Knowledge engineering is an area of active research and the theoretical basis of knowledge-based systems continues to develop as an aspect of artificial intelligence.

References

- A. Gonzalez and D. Dankel (1993). *The Engineering of Knowledge-Based Systems: Theory and Practice* (Englewood Cliffs, NJ, Prentice-Hall).
- B. Gaines, and M. Shaw (1993). "Knowledge acquisition tools based on personal construct psychology," *Knowledge Engineering Review*, Volume 8, No. 1.
- G. Kelly (1955). *Psychology of Personal Constructs* (New York, Norton).

Associated terminology: Artificial intelligence.

Knowledge management

Definition: Knowledge management is the process of creating, locating, encoding, and utilizing corporate knowledge to support and enhance corporate processes.

Overview

The idea of *Knowledge management* (KM) was proposed in the late 1990s as a mechanism through which organizations could capture their *tacit* and *explicit* intellectual property. Tacit knowledge is that information or knowledge that is not explicitly documented, primarily residing in the heads of the workforce. Explicit knowledge resides in some physical medium within the company, such as files, process manuals, and charts. The KM initiative was promoted in part by the corporate business process reengineering efforts that were also taking place during the late 1990s, which saw many firms downsize their workforce only to realize that the workers who were leaving were actually in possession of valuable assets: corporate knowledge, insight and "know how." To counter this, many organizations created a senior-level position of chief knowledge officer (CKO), whose responsibility was to create and utilize a knowledge base to enable and enhance corporate performance.

The CKOs quickly found that the process of capturing corporate knowledge was a difficult and resource-intensive activity, and the majority of initiatives were significantly reduced in scope. This resulted in the companies focusing upon key initiatives, amongst which were process-focused KM, workgroup collaboration, document management, human capital management knowledge bases, and corporate knowledge portals.

Process-focused KM is the creation of resources around the performance of a process. This may require knowledge engineers to perform knowledge elicitation upon domain specialists to determine how they perform their task. This is very similar in nature to the tasks associated with building a knowledge-based system or expert system, except that in KM the result may be a set of process manuals rather than a working knowledge-based system. Some process tasks are very amenable to the development of a knowledge portal. One such task is operating a call center, where an information system or knowledge portal can be developed to support the call-center operator. The portal would support them by providing scripts, easily accessible solutions to the most frequently asked questions, information resources that callers may request, hot links to transfer the customer to another level of support, and a knowledge base that contains solutions to documented problems. The system would also be self-monitoring, maintaining logs of calls so that improved solutions could be created in the future.

Workgroup collaboration involves providing support to project teams whose members may be located at different physical sites. The system would provide the ability to share documents and also allow members to share knowledge bases that support the processes and task being undertaken. Human capital management KM involves developing a corporate resource that identifies experts on a specific topic; these experts can then be drawn upon by other individuals as resources to help move a project forward.

Document management uses *Electronic document management* (EDM) software to capture, link, and provide version control for corporate data. Specialist EDM software overcomes the problem of corporate data often being in many diverse data forms and file types. The software allows any authorized member of the organization to search data that has been structured by the system. These are sometimes referred to as "enterprise knowledge portals."

Business value proposition

The use of KM systems allows corporations to capture and improve their process knowledge, share knowledge, and support knowledge-intensive tasks across workgroups. They also allow the structuring of archival corporate knowledge in an electronic-document system.

Summary of positive issues

Knowledge management software systems have been developed to support a variety of initiatives, ranging from specialist process tools (such as supporting call-center operation) to EDM systems that can capture and manage a wide array of types of corporate knowledge. The effective use of KM can increase and speed up access to relevant information by employees, ultimately reducing the costs associated with performing a task.

Summary of potentially negative issues

The KM initiatives are expensive and can require significant support over a long period of time. Knowledge elicitation and encoding in a suitable representation can be an extremely difficult and time-consuming task, as the knowledge engineering community can attest.

Reference

- P. Drucker, D. Garvin, L. Dorothy, S. Susan, and J. Brown (1998). *Harvard Business Review on Knowledge Management* (Boston, MA, Harvard Business School Press).

Associated terminology: Data-flow diagram, Knowledge-based system.

LAN/WAN/subnet/internet/intranet

Foundation concepts: Network, IP (internet protocol).
Definitions:

LAN: Local-area network.
WAN: Wide-area network.
Subnet: The group of computers served by a particular gateway.
Internet: The worldwide network of interconnected computers.
Intranet: An organization's internal network of computers.

Overview

A *Subnet* is a subset of a larger network consisting of a number of computers and other network access devices that have related IP (internet protocol) addresses, and are all connected to the larger network through the same network bridging device, or *Gateway*. The existence of subnets is an artifact of the way the internet protocol works; the broadcast network messages that comprise ARP (the Address Resolution Protocol, which allows IP addresses to be associated with particular hardware) will not be transmitted beyond a local subnet. However, the organization of computers into smaller-than-necessary subnets can be an aid to network administration.

A *Local-area network* (LAN) is similar in concept to a subnet, but less rigidly defined. A LAN is an organizational device that may consist of part of a subnet or a number of combined subnets. It is simply the portion of a network that is devoted to one organization or sub-organization, and generally has some uniform management.

A *Wide-area network* (WAN) is even less firmly defined. A WAN can simply be a group of connected or related LANs, most commonly the collection of LANs belonging to a single organization, or it could be any grouping up to and including the whole internet.

The term *Internet* is generally used to refer to the whole world of connected computers, the World Wide Web. It is also used as an adjective (as in "internet computing") to indicate communication between networks, as opposed to *Intranet*, which means communications within one network.

References

- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).
- D. Groth (2003). *A+ Complete* (Hoboken, NJ, Sybex–John Wiley and Sons).

Associated terminology: Virtual private network, Host, Web services.

Law cross-reference

In the United States, national or federal law relevant to computer systems is generally limited to protecting privacy, intellectual property rights, and computer systems owned by governments or financial institutions (although the Computer Fraud and Abuse Act does provide some protection to corporations). Other matters usually fall within the domain of state law, which of course means that there are more than 50 different versions currently in effect.

Those responsible for or dependent upon computer systems should always bear in mind that self-protection in the form of security and vigilance is the only real protection. No matter how strong and fiercely enforced national and state laws may be, they provide no protection against abuses originating from other nations. The idea of enforceable global laws against spam, child pornography, fraud, and software piracy is completely out of the question for the foreseeable future; there will always be lawless jurisdictions in which computer criminals may hide. So long as the law only forbids the origination of illegal materials, it will provide very little protection. Only forbidding internet service and telecommunications companies from relaying illegal material would have a real effect, and with current technology that would be exceptionally difficult.

United States federal law

This section provides no new information; it is a cross-reference to the articles concerned with individual computer-related laws. References of the form "XX USC YY" are to section YY of title XX of the United States Code (USC), where the most relevant parts of the law in question may be found. The USC is arranged under 50 titles indicating the general subject matter of the laws included; some laws fit under a variety of titles.*

Cable Communications Policy Act, 1984,
Public Law 98–549; 47 USC 521–551.
Protecting personal information collected by cable-service providers.

CAN-SPAM Act, 2003, discussed in *Spam* article,
Public Law 108–187; 15 USC 7701.
Enacts requirements on commercially motivated email.

Children's Online Privacy Protection Act, 1998,
15 USC 1301; 15 USC 6501–5606.
Restricting online data collection from minors.

Computer Fraud and Abuse Act, 1986,
Public Law 99–474; 18 USC 1030.
Forbids unauthorized access to certain computer systems, the release of harmful viruses, and trafficking in passwords.

Computer Security Act, 1987,
Public Law 100–235.
Requires the National Bureau of Standards to produce computer-security standards.

Digital Millennium Copyright Act, 1998,
Public Law 105–304; 17 USC 1201–1205.
Extends and strengthens copyright for digital materials.

Health Insurance Portability and Accountability Act, 1996,
Public Law 104–191; 42 USC 1320.
Enacts security and privacy requirements for digital medical systems.

Privacy Act, 1974,
Public Law 93–579; 5 USC 522.
Regulates the collection of personal data by the federal government.

Privacy Protection Act, 1980,
42 USC 2000.
Protects confidentiality of journalists' sources before publication.

Unlawful Access to Stored Communications Act,
18 USC 2701.
Guarantees privacy and confidentiality for electronic communications.

Voluntary Disclosure of Customer Communications or Records Act,
18 USC 2702.
Guarantees privacy and confidentiality for electronic communications and data stored by computing services.

Wiretap Act,
18 USC 2511.
Forbids tapping of telephones and other communications.

UK law

UK Computer Misuse Act, 1990.
Forbids unauthorized access to computers and the data held on them.

UK Data Protection Act, 1998.
Protecting personal information and restricting its transfer.

US state law

The varieties of state law make it impossible to provide any general coverage. We provide here references to two of the most significant computer-related laws for the state of Florida, simply as an illustration.

815.06, Offenses against computer users.

This act makes accessing any computer without authorization, disrupting computer services to authorized users, damaging

* Please note that the content of these and other law-related articles represents the views of the authors who are not legal experts, and the articles should not be read as presenting definitive legal advice.

computer systems, and introducing any "computer contaminant" (viruses, etc.) into a computer system a third-degree felony (5 years' imprisonment, $5000 fine), but if the illegal action results in damage valued at $5000 or over, furthers fraud, or interrupts a public service, it becomes a second-degree felony (15 years' imprisonment, $10 000 fine).

817.568, Criminal use of personal identification information.

This act makes the fraudulent use of another's personal identifying information without consent a third-degree felony, but that increases to a first-degree felony (30 years' imprisonment, $10 000 fine) if the fraud is valued at $50 000 or more, or if it involves 20 or more victims.

Legacy system

Foundation concepts: Software, Hardware, Architecture.

Definition: A legacy system is a systems architecture that does not support the functional requirements of an organization.

Overview

The term *Legacy system* comes from the fact that a great quantity of program code is handed down to programmers from previous generations of programmers; hence it is a legacy of the past. The term is generally used to indicate that a system is old, but this is an over-generalization. A system needs to be considered in terms of its ability to support the current and future processes of an organization; an inability to support changing process requirements is now taken as the definition of a legacy system. The total architecture needs to be considered in respect to the alignment of process requirements and requires an assessment of the components of the system's architecture: the software, the hardware, and the network components.

Business value proposition

Assessment of a corporate system's architecture with respect to its ability to support the present and future processes of a company presents an opportunity to assess the point at which systems will need to be replaced or upgraded. Legacy systems can incur high maintenance and support costs and increased costs for the organization as a whole if the technologies employed cannot support changing process needs.

Summary of positive issues

There are few positive aspects to systems architectures that do not support changing processes other than the fact that those systems may be well known and understood. For example, a Cobol system that supports the purchasing functions may, in a best-case scenario, be completely specified, documented, tested, and maintained. However, maintaining such a purchasing system for a very large organization can be extremely difficult and a resource drain upon the organization. The ability to run such a system with a deep knowledge and resource base facilitates an orderly transition to a more flexible environment such as an ERP system.

Summary of potentially negative issues

Legacy systems might not support the architectural or process requirements of an organization. At best they will be expensive to maintain and potentially deny the company access to the required processes necessary to support operations. At worst the system could fail or not adhere to legal requirements (such as HIPAA compliance), and has the potential to put a company out of business.

Reference

- W. Ulrich (2002). *Legacy Systems: Transformation Strategies* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: HIPAA, ERP, Cobol.

Logic programming

Foundation concept: Programming language.
Definition: A style of programming based not on giving detailed commands, but on describing the relationship between inputs and correct outputs through logic formulæ.

Overview

The traditional *Procedural* or *Imperative* style of programming, practiced by almost all programmers in the commercial environment, is based on the notion that the programmer should first work out how to solve a problem, working out an exact *Algorithm* for it, and should then *Code* that algorithm as a complex of detailed instructions that the computer will blindly obey. This model of programming is so ingrained in programmers that it is often thought of as the definition of programming, so that no other way is even conceivable.

Logic programming is another way. It is well known to experienced workers in the areas of artificial intelligence, but relatively unheard-of and almost completely misunderstood by the vast majority of programmers. *Prolog* is by far the most popular programming language for logic programming, although others do exist, and Prolog and logic programming are often seen as one and the same thing. In Prolog, no algorithms are designed, and no instructions are given. Instead, some facts are stated, both facts about data and facts about the relationship between inputs and correct outputs. The Prolog system uses these facts, according to the ancient established laws of logic, to transform questions into answers. The programmer must, of course, take great care to ensure that facts are presented correctly, in a way that will allow the Prolog system to perform the transformations successfully; programming with Prolog requires significant specialized expertise.

A small example may help to understand the logic programming style. First, some very basic facts will establish the participants in the problem domain. We have two mice, a cat, and a dog:

```
is-a (mouse, jerry).
is-a (mouse, jerrys-nephew).
is-a (cat, tom).
is-a (dog, spike).
```

The syntax may seem strange, but that is in the nature of programming languages in general. The meaning is probably clear. Next, some more general facts about how the world works are stated. One essential motivation for this world is that cats eat mice, or, in the terminology of logic programming, if X is a cat and Y is a mouse, then X eats Y:

```
eats (X, Y):- is-a (cat, X),
              is-a (mouse, Y).
```

Again, the syntax is strange, but if ":-" is read as "if," and the non-parenthetical comma is read as "and," it should make sense. Other motivating facts are that dogs chase cats, everything chases anything that it would like to eat, and one thing likes another if that other chases away a third thing that is trying to eat the first:

```
chases (X, Y):- is-a (dog, X),
                is-a (cat, Y).
chases (X, Y):- eats (X, Y).
likes (X, Y):- chases (Y, Z),
               eats (Z, X).
```

and that, perhaps surprisingly, is a complete, albeit simple, Prolog program.

The program does not tell the computer *how* to do anything; it certainly does not embody an algorithm. The *Prolog interpreter* (the run-time system) knows how to select a sequence of facts to apply in a given situation, and how to search for a path to a solution. It employs two essential techniques that are fundamental to logic programming: *Unification*, which allows it to work out which rules are applicable, and replace variables like "X" and "Y" with solid objects like "tom" and "jerry"; and

Backtracking, which allows it to try out one possible path in a computation, but abandon it if it does not work out, and try a different path instead.

A user runs the program by asking questions. A question may be a simple inquiry about a fact, such as "is tom a cat" or "does spike eat tom," and the system will reply with "yes" or "no," thus:

```
?- is-a (cat, tom).
yes.
?- eats (spike, tom).
no.
```

A user may ask a more general question by asking about a fact with a variable in it. The system finds all possible valuations of the variable that would result in the fact as typed being true. We may ask "who is a mouse" and "who eats jerry":

```
?- is-a(mouse, X).
X = jerry;
X = jerrys-nephew;
no more solutions.
?- eats (X, jerry).
X = tom;
no more solutions.
```

The computations work just as well in the reverse direction, a feature that is never available in traditional programming languages. For example, "whom does spike chase":

```
?- chases (spike, X).
X = tom;
no more solutions.
```

Questions may even have multiple variables, as in "tell me everything you know about who likes whom":

```
?- likes(X, Y).
X = jerry, Y = spike;
X = jerrys-nephew, Y = spike;
no more solutions.
```

Questions may also involve multiple parts, as in "find somebody who likes jerry and either is eaten by tom or chases spike":

```
?- likes (X, jerry),
   (eats (tom, X);
    chases (X, spike)).
no solutions.
```

Of course, nobody likes jerry. This is only a very basic example, but illustrates the style of logic programming. Logic programming in general, and Prolog in particular, opens up a whole new field to simple programming and rapid prototyping. Everything that can be done in a "normal" programming language can be done in Prolog, and conversely everything that can be done in Prolog could be done with a traditional programming language, but in most cases it would be a very complex and error-prone endeavor.

Business value proposition

Logic programming systems have been used extensively in the areas of artificial intelligence (AI) known as *Knowledge engineering*, *Intelligent tutoring systems*, and *Natural language processing*, since this approach to programming allows domain knowledge to be represented easily and facilitates the creation of rule sets that can operate upon that knowledge base.

While logic programming languages such as Prolog have been in existence for over 30 years, their adoption has not been widespread. Logic programming is almost exclusively limited to AI applications and use as a prototyping mechanism. However, the technology continues to evolve and the suitability of the approach to large projects has been enhanced by newer versions of Prolog with GUIs that are more aligned to "office"-type users, and by the provision of development environments to assist programmers in rapidly developing systems.

Summary of positive issues

Logic programs in some domains are easy to develop and do not suffer from many of the problems associated with procedural programming. Logic programs are easy to maintain and modify. There are many free

open-source implementations of logic languages available, and there are many tools and development environments that support this approach to programming. The documentation associated with logic programming is extensive.

Summary of potentially negative issues

Logic programs that solve traditional commercial problems can be difficult to write and database handling can also be difficult due to the fact that many implementations have poor input and output facilities. Very few programmers are trained in the logic style of programming. Logic programs can be difficult to interface with other systems, and logic programs are usually much slower than equivalent programs written in the traditional style.

Reference

- W. Clocksin and C. Mellish (1984). *Programming in Prolog* (Heidelberg, Springer-Verlag).

Machine learning

Foundation concept: Artificial intelligence.
Definition: Machine learning is the ability of programs to make inferences and expand their understanding of the domain in which they operate.

Overview

The term *Machine learning* is usually associated with automated learning or *Self-programming* capabilities rather than the production of solutions from static knowledge bases. Mechanisms for learning under current research include learning by knowledge acquisition, learning by examples, neural networks, case-based reasoning (CBR), genetic algorithms, and learning by discovery.

Knowledge acquisition is a vital component of machine learning and is the mechanism through which a system gathers knowledge, which then has to be represented internally to provide the basis for inference. Major challenges to researchers into machine learning lie in each part of this process: how to capture knowledge of different types (from pictures, speech, text, diagrams), how to represent each of these knowledge types, how to integrate them all, and, of course, how to infer useful new knowledge from old. In light of the multitude of different types of knowledge that systems need to acquire and be able to infer from, many techniques have been developed to assist in this process, including computational learning theory, explanation-based learning, delayed reinforcement learning, temporal difference learning, and using version spaces for learning.

One of the key problems still facing researchers is that of common sense reasoning; whereas a child having once touched a hot-plate can extrapolate that anything glowing red is potentially dangerous, this ability to generalize an experience is very difficult for programmers to build into computer systems, because every generalization is context dependent and thus variable in nature. One researcher examining this problem is Douglas Lenat, who created the Cyc (pronounced as in "psychology") project in 1984 in order to understand and collect common-sense knowledge (for example, we all know that we pour tea from a tea pot into a cup in order to make a cup of tea, but that depends upon the knowledge that gravity pulls the water downward and that concave objects like cups are capable of holding fluids). Much of what is considered to be common-sense is in fact a very complex web of uncounted essential facts that are usually learned through many years of continual experience. Capturing all of this implicit knowledge in computer-usable form is an exceptionally difficult task that has not been completed.

Business value proposition

Machine learning seemed to hold a significant degree of promise for several decades, but has only recently started to deliver techniques that help to solve business problems. These techniques are generally task-specific, for example, biometric identification techniques that include facial recognition or signature analysis through pattern recognition, and automated telephone-based customer relationship systems using machine learning to improve their understanding of speech and enhance their ability to perform natural language processing.

Neural network technologies have been used widely in diverse tasks such as stock market analysis, image recognition, and extrapolation from examples. Neural networks are mathematical programs that are trained with a set of known inputs and outputs, and adapt themselves to produce the correct sample output whenever presented with one of the sample inputs. For some types of problem, they are also able to provide correct solutions for inputs that did not appear in the training set. Neural networks are used in systems such as character

recognition, automomobile warranty analysis, and detection of credit-card fraud.

Summary of positive issues

Machine learning has a long academic history and some of the techniques have recently become sufficiently mature to be embedded in consumer and military products. Machine learning tools resulting from academic projects exist and are also available from commercial vendors.

Summary of potentially negative issues

Many techniques associated with machine learning are still difficult to implement in large-scale industrial-strength systems; they are better suited to less wide-ranging problems with limited domains. Some techniques such as neural networks are incapable of explanation and therefore the integrity of the solution produced may be difficult to verify.

References

- T. Mitchell (1997). *Machine Learning* (New York, McGraw-Hill).
- C. Matuszek, M. Witbrock, R. Kahlert, J. Cabral, D. Schneider, P. Shah, and D. Lenat (2005). "Searching for common sense: populating Cyc from the Web," in *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference*, Pittsburgh, Pennsylvania, ed. M. M. Veloso and S. Kambhampati (Cambridge, MA, AAAI Press/MIT Press), pp. 1430–1435.

Associated terminology: Neural networks, Knowledge-based systems.

Maintenance

Definition: Maintenance is the work performed to ensure that systems continue to perform to specification.

Overview

The activity commonly known as *Maintenance* is fundamentally the set of processes performed in order to ensure that a system performs to specification and will continue to do so. This term is broadly used to cover a variety of cases.

(i) *Proactive maintenance* ensures that the quality of a system's performance will be at or above the minimum service level requirements. This type of maintenance may involve both hardware and software processes, e.g., adding an extra server to handle a forecast increase in peak system demand by users.

(ii) *Reactive maintenance* occurs when a systems failure or emergency threatens the functionality of the system. Again this can be hardware- or software-related or a combination of both, e.g., a database hard drive crashes and the data needs to be recovered.

(iii) The traditional use of the term maintenance in relation to software has been to describe the work a programmer does to make changes or additions to existing software, e.g., to change the layout of data on a screen or to add a new section to a generated report.

Business value proposition

The maintenance function is one that all organizations have to undertake in one form or another. Proactive maintenance generally provides a positive return on investment but requires that organizations establish and develop metrics, and are proactive in using them. Without a set of baseline performance metrics, the benefits of proactive maintenance can be difficult to assess. Metrics are also established to determine the complexity of the code being

developed, since this will then help the IT organization not only to determine the cost and efforts required to perform maintenance but also to establish better processes and development methods to reduce code complexity in the future, and hence make the code more maintainable.

The establishment of a software quality assurance program helps to ensure that proactive maintenance prevents the need for reactive maintenance. Reactive maintenance is expensive and generally problematic because programmers and systems analysts are under pressure to perform a "quick fix" in order to re-establish services, and might not be allowed sufficient time to determine the true underlying cause of the problem. Endlessly fixing the symptoms of an undiscovered software or hardware fault can be very expensive.

With the move toward configurable systems such as ERP systems, there has been a decline in the amount of traditional maintenance activities, such as modifying legacy Cobol programs, and this has freed programmers to work on more critical core systems development.

Software maintenance will never disappear, since systems are always being modified or upgraded due to changes in the regulatory or technical frameworks within which they operate.

Summary of positive issues

Maintenance is a well-researched area of software engineering. Consultants and contract programmers are available to support a wide range of maintenance requirements, including the maintenance of legacy systems that are written in languages long thought to be extinct.

Summary of potentially negative issues

In order to prevent maintenance from becoming a never-ending and very expensive activity, it is necessary to undertake the expense of setting up a quality assurance program that establishes baselines and metrics. Companies can then look at processes by which to migrate their systems to environments that require less, easier, or cheaper maintenance.

Maintenance programming is not generally considered a popular career path for systems professionals, who would rather be creating new systems than modifying old ones. Some forms of maintenance programming are inherently dangerous: some systems that have been in existence for over 30 years will have been modified extensively during that period, and there might be no surviving overall structure, nobody who understands the existing system, and certainly nobody who understands how all the previous quick fixes might interact. A mistake in the maintenance process may lead to complete systems collapse.

Reference

- *The Journal of Software Maintenance: Research and Practice* (New York, John Wiley and Sons).

Associated terminology: Cobol, ERP, Software metrics.

Memory

Foundation concepts: Central processing unit, Storage.

Definition: The physical components of a computer system that support long- or short-term retention of information.

Overview

The need for memory in a computer system is absolute. Memory is needed in order to keep the sequence of commands that the computer is currently obeying (i.e., the "code" for the program or application itself) and the data that it is working on. All other programs that are available to the computer must also be stored so that they

can be executed on demand; similarly, all data that those applications could access must be kept available.

In human terms, the distinction between intellectual effort (processing or computing) and remembering (using memory) is difficult to see: we have a single organ, the brain, that performs both tasks. Indeed, according to current theories in neuroscience, there *is* no difference between the two tasks. In a computer system, there is a sharp distinction that must be understood.

If a computer is capable of performing 2 000 000 000 operations per second (which is not an unrealistic proposition), that means that it has a *Central processing unit* (CPU) that is capable of obeying 2 000 000 000 instructions per second. Those instructions have to come from somewhere: the CPU does not make its own decisions about what to do. Those instructions are the program or application that is being run, which was previously created by a human programmer, and is kept in the computer's memory. The computer's memory component must work at incredible speeds: 2 000 000 000 times per second, the CPU will effectively ask it "what shall I do next?," and expect to be given the next instruction to obey.

In addition to this requirement for exceptional speed, there are other conflicting demands. Absolute reliability is essential: a single wrong step in a program invalidates everything it is doing, and an error rate of just one in a billion would result in two failures every second.

Long-term retention of data is another essential. Corporate records and databases must be kept safely for an unlimited amount of time. Computers must have memory that continues to work even when the computer itself is turned off or damaged.

The fourth requirement is for capacity. If a company just has 25 000 customers, has an average of only 100 interactions with each customer (sales, purchases, enquiries, etc.), and records just a few lines of information for each interaction, the total amount of information that must be kept is already 100 times the size of the Bible. A large, active corporation could easily generate trillions of pieces of information.

Current technology is capable of satisfying all four of those requirements, *but not at the same time*. The main conflict is between speed and capacity. Small amounts of incredibly fast memory are affordable and reliable. Vast amounts of relatively slow memory are affordable and reliable. Vast amounts of incredibly fast memory are not just unaffordable, but also technologically infeasible.

To satisfy all of the requirements, a computer system has a complex memory structure, consisting of many layers of different technology working together to cover all of the requirements, as listed below.

CPU registers: a few dozen bytes of memory, physically part of the CPU, and working at the same speed as it, holding just the immediate details of one step of the computation.

Cache: around a megabyte of exceptionally fast memory used to hold temporarily just the portion of an application that is currently running and the portion of the data that it is actively processing; this is copied from slower memory as and when it is needed. Part of the cache (called "L1") is usually part of the CPU chip, and another part (called "L2") is usually an extra chip soldered onto the motherboard.

Main memory: hundreds, or even thousands, of megabytes of fast memory. This usually holds the entire running application and all data it is likely to make use of in the near future. Applications are copied from slower memory into main memory when execution begins. Main memory, cache, and CPU registers are

volatile, meaning that their contents are lost when the computer is turned off. Main memory (sometimes called RAM) is usually in the form of one to four replaceable components (DIMMs, SIMMs, etc.), which are plugged into the motherboard.

Disk: hundreds, or even thousands, of gigabytes of slow memory. Disks are used only for long-term storage, or when applications need more data than can fit into main memory. Disks (often called *hard* disks) are non-volatile, meaning that they retain their information even when the power is turned off.

Backing storage (for example, DVDs and CDs): very slow memory, but of unlimited size, since the media (individual disks) can be swapped while an application is running.

Main memory may be literally a million times faster than disk, but is also around a thousand times more expensive, and it is also volatile. So a real-world computer system can not reasonably be made faster by using entirely main memory instead of disks, or cheaper by using entirely disk instead of main memory.

Disks may be around 100 times faster than DVDs, but, whereas DVDs can be exchanged in a matter of seconds, changing a hard disk is a fairly major and slightly risky operation, to be performed only when the computer is off.

Portable memory devices based on *Flash-memory* technology provide another alternative. These small devices are easily portable, and can be plugged into almost any computer, through its USB port, without any preparation. They can currently store up to a few gigabytes (although that figure is still increasing), but, due to their slow speed, they can be considered only as data-transport or backup media, and are certainly not an alternative to "main" memory or RAM.

Business value proposition

Memory is a critical component and significantly impacts the performance and reliability of the whole computer system. Generally, the more memory at each of the levels, the higher the performance of the system up to a limit set by the ability of the processor to perform its tasks. Generally, the performance of a computer system can be significantly improved by the addition of memory above and beyond the minimal configuration requirements of the system. A significant proportion of untraceable problems will just "disappear" if more memory is made available.

Summary of positive issues

The decreasing cost associated with memory at all levels has been one of the driving forces behind the production of faster and more robust systems. Memory devices are available in many types and configurations, internal to the computer, removable, redundant backup, and memory sticks to name just a few. This provides a high degree of flexibility in the design of systems and architectures.

Associated terminology: Memory and disk size, Disk.

Memory and disk size

Overview

A computer's memory capacity, just like that of a human being, is finite. If an application is required to "remember" more things than its memory can hold, it will simply not work. Unlike human memory, the size of a computer's memory is both exactly quantifiable and changeable. If more memory is needed, it can be bought and plugged into the motherboard in a matter of a few minutes (subject to a maximum limit imposed by the computer's design).

The total size of a computer's memory is measured in terms of how many pieces of

data it can record (this corresponds to the length of the memory), and how big each of those pieces of data can be (corresponding to its width). A computer that can store a million ten-digit numbers has more capacity than does one that can store two million three-digit numbers.

Since the beginning of the computer era, the width of memory, the size of an individual data item, was always called a *Word*. Different kinds of computer would have different word sizes (e.g., ICL in the 1970s frequently used 24-bit words, giving 7-digit data items; DEC/PDP in the same period frequently used 36-bit words, giving 11-digit numbers). Thus memory size was always quoted in terms of how many words there are and how big a word is (e.g., 49 152 24-bit words for an ICL-1902). The word *Byte* was always used to refer to some portion smaller than a whole word of memory.

Since the supremacy of the desktop computer the terminology has changed. In common usage today, the word *byte* exclusively means exactly eight bits (enough to store a two-to-three-digit number) and *word*, if it is used at all, just means two bytes. The amount of memory in a computer is always given simply as the total number of bytes, with no reference being made to the width. This simplifies comparisons, but hides useful data: a computer with 64-bit-wide memory will generally be much faster than a similar computer with 8-bit-wide memory.

Since a computer usually has a very large amount of memory, special abbreviations are used for large multipliers; these are K, M, G, and occasionally T. There is a great deal of confusion and a certain amount of deceit involved in the use of these abbreviations.

The abbreviation "K" is pronounced simply as "kay"; it is *not* "kilo." In memory capacities, a capital "K" is always used, and the multiplier that it represents is 1024. A computer with 4 KB ("four kay bytes") has 4096 bytes of memory. The word *kilobyte* is almost invariably a mistake. The metric or S.I. prefix *kilo* (as in *kilogram* or *kilometer*) is always written as a small "k," so there should be no confusion.

From a design point of view, the idea of a computer with 1000, 1 000 000 or 1 000 000 000 bytes of memory is absurd, whereas 1024, 1 048 576, and 1 073 741 824 bytes would be perfectly reasonable sizes. In computer technology 1024 is a "nice round number," just as 1000 is a nice round number to humans. Memory capacity is always some multiple of 1024 bytes, and, with 1024 being so close to 1000, the letter "K," being so close to the metric prefix "k," was an obvious choice. 12K is an easy number to remember; 12 288 is not. Originally, when only technically proficient people ever discussed memory capacity, there was never any confusion.

The other prefixes, M, G, and T, have a less clear standing. They were "borrowed" directly from the metric system, where M for "mega" means 1 000 000, G for "giga" means 1 000 000 000 and T for "tera" means 1 000 000 000 000. When somebody talks about "one megabyte" or 64 MB of memory, what do they mean? Although "mega" very clearly means "one million," no computer ever had one million bytes of memory. What is meant in these cases is 1 048 576 (i.e., 1024×1024), which is a standard unit of memory size.

- A "megabyte," 1 MB, is most commonly 1 048 576 bytes.
- A "gigabyte," 1 GB, is most commonly 1 073 741 824 bytes.
- A "terabyte," 1 TB, is most commonly 1 099 511 627 776 bytes.

Some manufacturers take advantage of this unfortunate naming to exaggerate the capacity of a disk drive by using prefixes from the metric system in a context in which the computer-technology prefixes are obviously expected. For example, if a disk has a capacity of 3 221 225 472 bytes, an honest label could say "3 GB,"

"3072 MB," or even "3 221 225 472 bytes." A deceptive label might say "3.22 GB."

The difference seems too small to worry about: 3.22 GB is only just over 7% more than 3 GB; but the problem can be significant. $10 is just one tenth of one percent more than $9.99, but the psychological effect of the price difference is well known. When confronted with a choice between a 3.22 GB disk drive from an unheard-of manufacturer and a 3 GB disk drive from a reputable one, many purchasers opt for the 3.22 GB drive because it seems to be better value for money, even though it is in fact exactly the same in terms of capacity, and may well be significantly worse value in terms of reliability.

Business value proposition

Read the label carefully. A difference of a few percent in disk capacity will have no significant effect, but a difference in reliability is the difference between successful productivity and ruinous disaster.

Middleware

Foundation concept: Software.

Definition: Middleware is code that is used to enable two or more other applications that would otherwise not be able to communicate to do so.

Overview

The term *Middleware* is used to denote software that enables two other software systems to communicate and pass data from one to the other. The need for middleware became pressing in the 1970s, when organizations commonly ran different applications on different machines made by different manufacturers, that each had its own proprietary operating system, database, and network-communication protocols (a heterogeneous environment). Standardized, system-independent network protocols had not yet been developed, so these machines were not capable of communicating without additional software to act as an intermediary.

Another form of middleware is known as a *Front-end processor*. This is software, or even special-purpose hardware, that interfaces a system with the outside world, perhaps reformatting or condensing inputs, or perhaps converting output into a more human-friendly form. Front-end processors are a variety of middleware that makes a single system accessible, rather than interconnecting two systems.

Middleware requires specific information about the protocols used and the formats of the data being received from each system. The development of middleware has traditionally been a complex task, further complicated by the need to have different middleware solutions for each partner computer that might ever be connected. Then, when a data format, operating system, or communications protocol changed, the middleware would also have to be changed; if many different computers were involved, each change would represent a major expense.

Many of the problems associated with enabling computers and applications to communicate that arose during the pre-internet era have subsequently been eased or resolved. It should be noted that the term "middleware" was not commonly used until the late 1980s, when distributed corporate systems architectures started to evolve. Prior to this the term "systems integration" tended to be used.

A primary enabler of easier communication has been the adoption of standard internet protocols such as TCP/IP, which provide well-defined languages and media through which computers can exchange data. The development and adoption of XML, which defines a uniform format for all kinds of data, has freed developers from having to write customized data-conversion software for every entity with which their system communicates.

The problem of inter-operability has also been addressed by the *Object Management Group* (OMG), whose members focus upon producing and maintaining specifications for *Inter-operable enterprise applications*. Their best-known product is the *Common Object Request Broker Architecture* (CORBA) specification, which provides a specification (through an interface definition language, IDL) that allows developers to define interfaces for their programs using a standard language that can be understood and used by any IT organization.

Business value proposition

Middleware has always been used to enable computer systems to communicate with each other. The term *Systems integrator* is used for consultants who build middleware to interconnect disparate systems. An example of this is when an organization pursues a "best-of-breed" strategy for an ERP implementation. In such a project, a company may buy an HRM system from one company and a financial system from another. In order for the systems to communicate effectively with each other, the system would potentially require some middleware to be written. This would require, on the part of the system integrators, specific information about the packages, including the protocols used and the data requirements of each. The process of integrating modern systems is simplified when vendors adopt open standards and protocols such as TCP/IP and XML. It is further aided by vendors' adoption of the CORBA interface definition language, or a similar technology, so that a system's interface requirements can be clearly understood.

Summary of positive issues

Middleware techniques are well known and supported within the IT community. The Object Management Group has created the CORBA specification, which provides a specification (the interface definition language, IDL) that allows organizations to create middleware in line with a standard methodology and specification system.

Summary of potentially negative issues

The development of middleware can be a difficult and resource-intensive operation, and the standards, particularly CORBA, may be very complex.

Reference

- P. Bernstein (1996). "Middleware: a model for distributed system services," *Communications of the A.C.M.*, Volume 39, No. 2.

Associated terminology: ERP.

MIS (management information

Definition: The management information systems department is the functional entity within a corporation that is responsible for information systems and the alignment of those systems with corporate strategy.

Overview

In the context of a business, the MIS department or "IT organization" is the functional area tasked with providing the business enterprise with the technology resources necessary to perform its functions effectively. The MIS department, through the office of the chief information officer (CIO), provides advice on technological strategy to the board of directors and senior executives of the company, while maintaining current systems, developing systems to meet future needs, ensuring regulatory compliance, developing disaster contingency plans, ensuring security, and supporting and enabling the business units and divisions.

In the context of academic study, the MIS department is usually located within the School of Business and focuses upon providing business-related information systems skills to its students. This differentiates

MIS departments from computer science departments, which focus upon the theoretical aspects of computing and programming systems, and computer engineering departments, which tend to focus upon the technological aspects of systems.

Business value proposition

Corporate MIS departments emerged in the mid 1980s, evolving from *Data processing departments*. The role of MIS departments has been transformed in many ways and can be considered to reflect what Venkatraman terms a *Value center* rather than a *Cost center* as they were previously considered to be.

A value-centered MIS organization is based upon four concepts. Firstly, rather than being a cost center the MIS department provides a technological environment that not only supports corporate processes but also enables those processes to achieve high operational efficiency. Secondly, the MIS department also acts as a service center that provides "drivers" of competitive advantage to the organization. A typical example is to think of the help desk as a cost center, but in the value-centered approach it is the contribution of the help desk to the overall performance of the enterprise that is considered, not its absolute cost. The third dimension of the value-centered MIS department is its ability to act as an investment center that attempts to derive new high-yielding business opportunities from existing IT resources and emerging technologies. The fourth aspect of the value-centered approach is to consider the MIS department as a profit center that provides services and products to external markets at a profit.

The MIS department is tasked with many responsibilities in the modern organization, ranging from providing advice on technological strategy to the CEO and Sarbanes–Oxley compliance information to the board of directors, to the development of mission-critical software systems that form the very core of the business' competitive strengths. The MIS department is also tasked with ensuring systems integrity and security, maintaining and upgrading current systems, assessing outsourcing options, working with software vendors, developing systems directly with customers and suppliers, complying with regulatory agencies, developing disaster contingency plans, and supporting and enabling the business units and divisions.

The study of MIS occurs in the MIS departments of Business Schools where scholars undertake research into the multitude of activities associated with corporate systems. There are over fifty scholarly journals that publish the research of the academic community and several professional organizations and societies that promote scholarship, education, and standards in the professional IT community.

References

- N. Venkatraman (1997). "Beyond outsourcing: managing IT resources as a value center," *Sloan Management Review*, Spring.
- *MIS Quarterly*.
- *Communications of the A.C.M.*
- *Information Systems Research*.
- *Journal of MIS*.
- *Information & Management*.
- *European Journal of Information Systems*.
- *Journal of Information Technology*.

Associated terminology: Chief information officer, Enterprise resource planning.

Modem

Foundation concepts: Binary, Bit, Analog, Bandwidth.
Definition: A device that converts digital data signals into audio form (and vice versa), enabling transmission on telephone lines.

Overview

The word "modem" is a contraction of *Modulator–demodulator*. *Modulation* is the act

of modifying one signal, the *Carrier*, so that it carries another signal with it; *Demodulation* is the reverse: extracting a signal that was carried on the back of another.

With a modem, the carrier is an audio signal, an audible whistle, and the signal carried is any piece of digital data. All data may be encoded as a simple sequence of 1s and 0s: binary. A sequence of 1s and 0s can easily be used to control an audio signal by varying its pitch. A 1 is represented by one note, and a 0 by another. The electronic circuitry required to perform this kind of modulation is extremely simple and cheap. Demodulation, namely detecting which note is being received, and converting it back to a 1 or 0 as appropriate, is similarly simple and cheap.

In essence, that is all a modem is. Digital data is converted into a series of musical notes and back again. The reason for this is, of course, the desire to make computers communicate over long distances. Setting up a large-scale digital network is a very expensive and time-consuming task. The public telephone network was already in existence, and reached almost everywhere, so it provided an obvious solution. Telephone networks can transmit only audible signals; purely digital data will simply not get through, so the modem is the essential intermediary, converting digital data into a form that can be transmitted on a standard unmodified public telephone network.

Public telephone networks apply bandwidth limitation to all transmitted signals. The highest frequencies are cut off, which is why the letters S and F can be hard to distinguish (the difference is in the details of the high-pitched hiss), and dog whistles are completely cut off. The reason is simple economics: with bandwidth limitation, many signals can be multiplexed to share the same long-distance lines.

If the carrier signal has a limited bandwidth, then so does the carried signal. This is one of the most commonly misunderstood but most basic facts of information theory. If the carrier signal is cut off at 5 kHz (quite a high note really, in musical terms), then no clever inventions will get more than 5000 bits of information through per second. High-speed modems rely on the fact that data and information are not quite the same thing: the data that people want to transmit usually has a high degree of redundancy, and can easily be compressed into something smaller: 50 000 bits of personal data could be quite easy to compress into 5000 bits of real information.

The use of dial-up service over a normal telephone line requires a modem at both ends. Broadband connections (DSL, cable modem, etc.) use devices that are called modems, and they do modulate and demodulate signals, but they are not using a standard unmodified telephone line; so they are not subject to the same bandwidth limitations. Direct network connections (ethernet, fiber optics, etc.) use devices that are never called modems, even though they do perform the same tasks of modulation and demodulation outside the realm of audio signals.

Business value proposition

Modem technology has been an integral part of computer networks for decades, and today it provides high degrees of reliability at low cost. Modems range from external devices through which any computer may be connected to an external network to the built-in devices typical of laptops, and provide a series of options for systems managers to select from. Wireless and other network connections use a device that is technically a modem, but almost never so called.

While traditional modems are in general terms slower than broadband modems, they are still sometimes useful as a mechanism for interactive situations that require a consistently low degree of latency in the response. A common example occurs when connecting to a command-line system such

as Unix system through a *Telnet* client. A broadband connection may go through many dozens of routers, and produce a significant delay in feedback. A delay of half a second when downloading web pages or transferring files is barely detectable, but for an inexpert typist a delay of half a second between pressing a key and seeing the letter appear is disastrous. Dial-up modems use circuit-switched connections, providing a direct router-free path, and feedback delays are virtually nonexistent. The lower bandwidth is irrelevant for simple human-speed character-mode communications. Modems also provide an additional but small level of security: dial out can be disabled, and caller identification may be used to ensure that only authorized accesses are possible.

Modems also provide systems managers with a mechanism for remotely dialing into a system to perform corrective or maintenance activities should they be required. Telephone systems are not directly powered by the public electricity supply, and usually remain operational during blackouts, so a systems administrator with a battery-powered laptop can still connect to central systems during emergencies.

Summary of positive issues

Modems are reliable devices with relatively low costs associated with their purchase and maintenance; they provide a valuable secondary means of access to systems in emergencies, or when low bandwidth is acceptable and fast latency is required.

Summary of potentially negative issues

Modems are limited in speed by the bandwidth provided to them by the public telecommunications companies which carry their signals. Some characteristics typical of broadband connections may be detrimental to certain activities, such as connecting to remote computers over a Telnet connection.

Reference

- M. Banks (2000). *The Modem Reference: The Complete Guide to PC Communications* (New York, Cyberage Books).

Associated terminology: Bandwidth, Cable, and connectors, Voice over IP.

Motherboard

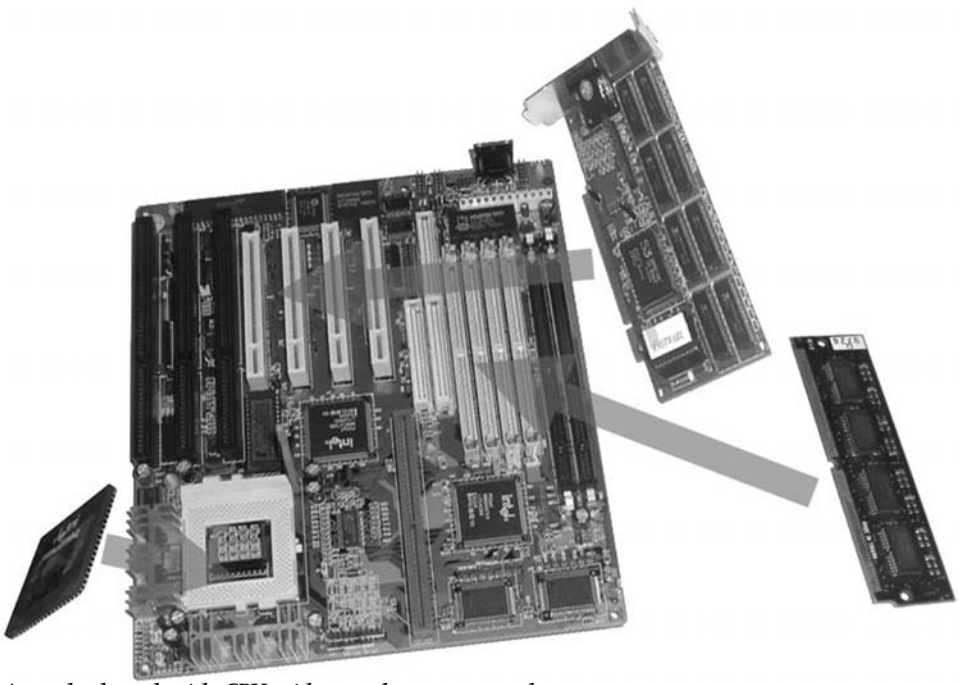
Definition: An essential circuit board on which CPU, memory, and required support chips are mounted, which also provides connectors for auxiliary or optional components, and forms the backbone of a minicomputer or microcomputer.

Overview

Even though the fourth generation was heralded as whole computers on a single chip, no general-purpose mainstream computer actually exists on a single chip. It is not possible to take (for example) a Pentium 4 chip, connect it to a keyboard, monitor, and power supply, and expect it to work. Of course, one can *expect* it to work, but it won't.

CPUs still require a large amount of external support circuitry just to achieve the basic required functionality of a computer. Required components such as memory, disk controllers, clock generators, etc. need intricate carefully designed connections between themselves and the CPU. The motherboard is simply a large printed circuit board with all the support components soldered in and empty sockets ready to receive the CPU and memory chips. The CPU and memory are not fixed parts of the motherboard because there are too many buyer-selectable options. There are many types and speeds of CPU, and many sizes and speeds of memory that could be used with one uniformly designed motherboard.

Additionally, a motherboard has a number of vacant connectors ready to receive extra "optional" components. Originally,



A motherboard with CPU, video, and memory cards.

disks, graphical displays, printer ports, and network connections were considered to be optional components. They were purchased as separate items and plugged into the motherboard's sockets. As those components gradually became regarded as essential and the potential for miniaturization increased, it became more and more common to provide them as fixed parts of the motherboard. It is now quite commonplace for a personal computer or workstation not to require any additional circuit boards beyond the motherboard. This makes computers cheaper to purchase, but does have disadvantages: once a motherboard component fails or becomes outdated, it can not be replaced unless the whole motherboard is replaced.

Business value proposition

For many, motherboards are not a consideration. Off-the-shelf ready-to-run computers are the most common purchase, and all decisions concerning the motherboard and CPU have already been made by the manufacturer. Mainframe computers generally do not have motherboards, but access to a mainframe computer will almost certainly be through a normal personal computer acting as a workstation.

For those who decide to save money by purchasing computer components and assembling them in-house, selection of the right motherboard is absolutely critical, probably more important than selection of the right CPU. There is an incredible number of motherboard manufacturers throughout the world, and some seem to offer incredibly good deals. The correct design of a motherboard is a very difficult procedure, and very small design flaws will render a computer totally unreliable. It is very difficult to make an objective assessment of the quality of a motherboard without using it extensively for a long period. Most motherboards never get print reviews, and it is often not possible to tell whether online reviews are written by real users and

disinterested third parties, or were planted or financed by manufacturers or retailers.

It is also essential to realize that CPUs come in different, incompatible families. An AMD Athlon (for example) will not fit in a motherboard designed for Intel Pentiums or Apple G4s. Motherboards also have a predetermined range of supported CPU speeds, so, although a 3000 MHz Pentium 4 may work in a motherboard designed for 1000 MHz Pentium 4s, it probably will not work at full speed. Memory components also come in different families, only one of which will be compatible with any given motherboard. This all means that upgrading a CPU often requires upgrading the motherboard and memory too.

Summary of positive issues

A uniform basis for computer construction allows the CPU, memory, and other components to be freely chosen and configured in a "mix-and-match" fashion. Damaged major components may be replaced with compatible new components without having to replace the whole computer.

Summary of potentially negative issues

Different families and speeds of CPU and memory create expensive compatibility problems, often making expected upgrades impossible. Essential components built into a motherboard can not be replaced if they fail.

Associated terminology: CPU, Memory.

Multicast

Foundation concepts: Network, Internet protocol.
Definition: A network transmission that is intended to have more than one recipient.

Overview

There are three different modes of communication amongst the members of an interacting network community. *Point-to-point* communications involve a direct transmission from one sender to one specific receiver, and this is how nearly all internet traffic is delivered. *Broadcasting* requires a common carrier medium, often called an *Ether*; a transmission is put out "on the air" by one transmitter, and everything on the network receives it. Many, perhaps most, receiving stations will choose to ignore the broadcast, but it is still available to all. Broadcasting is how radio and television signals are delivered, and *does not exist* over the internet. The third mode, *Multicasting*, is used when the effect of a broadcast is desired, and no common carrier medium exists.

There is no common carrier medium spanning the internet. This simply means that broadcasting is impossible: when one system transmits a message, it can be received only by other stations on the same local-area network (LAN); other systems have no connection to it. Long-distance transmissions (anything beyond the LAN) require repeated retransmission over individual network branches until the desired receiver can be reached.

If a system is required to send the same message to multiple receivers, it must either send multiple exact copies of that message, one to each recipient, or use multicasting. Sending multiple copies is clearly a very inefficient use of resources, especially network bandwidth, but it is still a common implementation choice, because multicasting is a very difficult technology.

True network multicasting comes in two varieties. The first is extremely easy and efficient to implement. If all recipients are on the same LAN, there is no problem. Broadcasting does work over LANs, so a message can be sent once, using the standard Layer 2 Protocol, typically Ethernet. The other variety is very complex. IP multicast, the only kind that applies to the existing internet infrastructure, requires that systems adopt a special shared "multicast IP address"; the transmitter sends a

message to that address, and all routers and gateways must know to forward messages with multicast addresses in multiple directions. This requires a lot of cross-network organization; most internet routers do not support multicasting. True multicasting can be relied upon only in a controlled network environment, perhaps a corporate network in which all routers are subject to the same corporate policy. Without such a controlled and friendly environment, out-of-LAN multicasting is a very complex and inefficient technology.

Business value proposition

Multicasting is a possible technique for distributing a communication amongst the members of an interacting network community. While the technology for internet multicasting beyond a LAN is still developing, multicasting within a LAN is a useful mechanism for distributing data. Multicasting provides an efficient and effective technique for corporate audio and video conferencing, and for online training over the same internal network.

Summary of positive issues

Multicasting is a useful, efficient, and effective technique for communicating amongst a group of users on a LAN.

Summary of potentially negative issues

Multicasting beyond the confines of a LAN is not at all well supported. It is still a developing research topic.

Associated terminology: LAN (local-area network), OSI seven-layer model, Ethernet, Internet protocol.

Natural language processing (NLP)

Foundation concept: Artificial intelligence.
Definition: The development and use of techniques to understand natural language input and to formulate an intelligent response.

Overview

Since the earliest days of computing, the idea of interacting with a machine through speech has been very appealing. To do so requires solving two problems: firstly, a system must be able to take a captured audio signal and recognize the words within it; and secondly, the system then has to be able to understand what the whole thing means. Researchers have been working upon both of these problems and have achieved a certain degree of success in each. It is not uncommon for business users to interact with airline reservation systems, for example, through an automated voice-recognition system. Many of the more successful systems bypass most of the first stage by accepting only typed input, not audible speech.

The origins of *Natural language processing* (NLP) are almost as old as computing itself. In the 1950s and 1960s researchers began to develop a variety of techniques aimed at understanding the ways computers could be used to provide NLP. One such system, *Eliza*, was created by Joseph Weizenbaum, whose system pretended to be a psychologist with whom a dialog could be enacted; a typical interaction with Eliza might start like this:

Eliza:	How are you?
Human:	I'm feeling ill.
Eliza:	How long have you been feeling ill?
Human:	Just a few days.
Eliza:	Do carry on.

And the dialog would continue. The secret to the system was that it used pattern matching, in that the response to the opening question would be matched with a set of input–output templates such as (I'm feeling X)–(How long have you been feeling X?). When a match was found for an input template, the corresponding output template would be printed. Together with a few simple rules for transforming inputs, such as You→I, My→Your, etc., and some stock responses for when nothing matches such as (Do carry on), that was the whole system. Eliza is exceptionally simple; it does not understand what the user is saying in any meaningful way, and it does not seem possible that such a system could be of any use, but Eliza-like systems proved very provocative when they were developed and many people were initially fooled by the interaction. The simplistic pattern-matching techniques used by the system place a serious limitation on how far the system's interactions can go. No real attempt is made by the system to understand the dialog, since the system progresses from one pattern-based response to another, without any memory of what was said before.

More recently, very sophisticated systems have been designed, which do seriously analyze user inputs, construct an internal representation of what they mean, and conduct directed meaningful conversations. Such NLP systems are a valuable interface tool for *Expert systems* and automated customer response systems. The computer processing of human languages is an exceptionally difficult task, as is the representation of meaning and the understanding of a train of thought; no system has yet come close to the levels of performance expected of any competent human.

Another early research area was in *Machine translation*, which attempts to automatically translate one natural human language into another. It is immediately clear that simplistic dictionary-based word-for-word translation does not work, because different languages have different grammatical structures and vastly different idioms. The only way to translate

properly is to perform full NLP to understand the grammatical structure of the input, and then reverse the process to convert the structure back to speech. Automatic translation is still an incompletely solved problem.

The ability to capture context was studied by Noam Chomsky, who formulated a new theory for the basis of linguistic research and computational NLP systems. His theory is based upon the use of *Generative grammars*, constructs used to describe how a sentence is formed: e.g., a [sentence] may be decomposed into a [noun phrase] followed by a [verb phrase], the [noun phrase] and the [verb phrase] may then be broken down into other constructs such as [determiner], [noun], [verb], [article], [adjective], [adverb], etc. These constructs may then be used to create formal grammars through which an input stream of words may be parsed as a first step toward extracting their meaning.

There have been many types of grammars used within natural language research, including *Phrase-structured grammars*, *Transformational grammars*, *Case grammars*, and *Augmented transition networks*, and parsing is a central task upon which NLP is based. Typically, commercial systems such as those used in customer-relationship management are built to cover dialogs in very limited areas, and the limited vocabulary helps simplify the processes of grammar construction, parsing, and response generation.

Business value proposition

Public perception of NLP has been very positive since the days of seeing Robby the Robot from *Forbidden Planet* and the tinny-voiced computer of *Star Trek* responding helpfully and in perfect natural language to every question posed by the crew. Even though reality still does not come close to those ideals, and shows no signs of doing so in the foreseeable future, the general public continues to believe otherwise.

One aspect of NLP technology that has been deployed in a variety of ways, with a variety of levels of success, is that of voice recognition. Voice recognition is an extension of NLP in which spoken words in audio format are taken as input, rather than typed text. This adds a whole new layer of difficulty to an already intractable problem: that of recognizing words, one of the many tasks that humans handle with ease but for which no reliable computational method has yet been discovered. This is not to say that the technologies associated with voice recognition have not been commercially exploited; indeed, specialist systems do exist. For example, word-processing systems that recognize the user's voice commands have been created, although considerable training of the system is frequently required when the system is to be used in domains with specialist vocabulary requirements.

Call centers use a technology known as *Interactive voice response* (IVR) to automate aspects of their customer service operations. These systems work in a dialog mode and, to be successful, are required to be very flexible in their voice recognition component due to the potential range of voices, accents, and dialects with which they have to interact. So far, they can successfully handle only the simplest of situations.

NLP is also used in systems that automate the reading of customer emails, attempting to interpret their desires and responding appropriately. An area of growth for NLP is that of computer-based training, for example for instruction in a foreign language, mathematics problems, or corporate processes.

Summary of positive issues

The academic literature on NLP and voice recognition systems is extensive. Many commercially supported systems are available for a wide variety of tasks. NLP systems are best deployed on automating routine

tasks, rather than ones that are potentially customer-sensitive in nature, and, used as such, can provide a positive return on investment.

Summary of potentially negative issues

Voice recognition and NLP systems have not yet reached the level at which they can interact naturally with human users, either in writing or vocally. Complex data mining or open-ended questions are generally not handled well, and levels of understanding are adequate only in narrowly specific domains. The training overhead associated with voice recognition systems can be high. Interactive voice response systems can have a negative impact upon customers' perceptions of an organization's level of service and commitment.

References

- A. Turing (1950). "Computing machinery and intelligence," *Mind*, No. 59.
- N. Chomsky (2002). *Syntactic Structures* (Berlin, Mouton de Gruyter).
- J. Weizenbaum (1966). "A computer program for the study of natural language communication between man and machine," *Communications of the A.C.M.*, Volume 9, No. 1.
- A. Barr and E. Feigenbaum (1981). *The Handbook of A.I.*, Volume 1 (London, Pitman).
- R. Plant and S. Murrell (2005). "A natural language help system shell through functional programming," *Knowledge Based Systems*, **18**, 19–35.

Associated terminology: Machine learning, Data mining.

Network

Definition: Any system of interconnected communicating computers.

Overview

A network is simply a group of computers and the equipment that connects them. It may range in scale from a home setup consisting of a desktop and a laptop that are occasionally synchronized, to the entire internet.

The purpose of a network is, of course, to enable communications, but there is great variety in what can be achieved by those communications. A network may be configured to make all workstations appear identical, so that a worker may go to any computer at any time, and find exactly the same files and applications, in exactly the same states. At the other extreme, a network may be configured so that every computer remains totally independent, and the only network traffic permitted is simple messages under explicit user control (such as emails and "chat" sessions).

Most networks fall somewhere between the two extremes, allowing individual users to control what is public and what is private on their own computers. Arbitrary parts of a file system may be made to appear to be shared across a network (perhaps using NFS, the *Network file system*, or shared folders), may be made available only to specific access requests (by setting up an FTP or HTTP/web server), or may be shared on a more cooperative basis using one of the many peer-to-peer file-sharing services.

Networks are generally not uniform organizations, but rather consist of a number of different structures at different levels. The entire internet is a single network, but is also a large number of interconnected sub-networks. The usual arrangement is for each organization to have its own LAN (local-area network) consisting of the computers it uses, interconnected with only the bandwidth (transmission capacity) required for their own use. Each computer on a LAN can communicate directly only with other computers on the same LAN. Except in the few cases where internet access is

not desired, each LAN will also incorporate a bridging device (sometimes a computer with two network cards, sometimes a special-purpose device such as a *Switch*, a *Bridge*, or a *Router*). The bridging device communicates both on the LAN and on the next level of network above. Any computer on the LAN wishing to communicate outside the LAN must have all communications forwarded by the bridging device. This simplifies network management (only one device needs to know how to deal with the internet as a whole), and provides a useful security bottleneck.

The bridging devices for a number of LANs may be similarly connected together on a larger-scale network, possibly built with higher-bandwidth components since it must carry all inter-LAN and internet traffic, and this larger network has its own bridging device that connects it to the next level of network. Ultimately, there will be a very-high-capacity connection to the entire internet *backbone*. LANs are most commonly built on ethernet technology, giving bandwidths from 10 to 1000 million bits per second. Higher-level connections use a much wider variety of technology to support the much higher capacities needed, including fiber optics, microwaves, and satellite communications.

The way in which computers are connected in a network is sometimes referred to as the "topology" of that network. These topologies take their names from the shape of the network and include *Bus*, *Star*, *Ring*, and *Mesh*.

A *Bus* topology consists of a single cable along which computers are connected in a long row. A computer communicates by putting on the cable a message that is read by all the computers, but acted upon only by the one with the correct address. This topology is effective only when there are only a few computers, since only one computer can put a message on the cable at any one time. The bus topology is, however, cheap and easy to build. This is typical of *Thinwire ethernet*.

A *Star* topology comprises a central hub or *Switch* into which a cable from each computer runs, so that a computer sends a message to the hub, which then relays the message to either all the computers in its network, or, in the case of a switch, only the correct one. This configuration is easy to build and is resilient unless the hub has a problem. Most modern LANs have a star topology.

A *Ring* network, sometimes called a *Token ring*, is a network of machines connected in a circle. Messages are sent in one direction around the ring, from one computer to the next, until a machine recognizes itself as the intended recipient. If the message makes its way all around the ring, then the original sender recognizes it and the message is recognized as undeliverable. To control proper sharing of network bandwidth, a dummy message called the *Token* is passed around the ring when there is no real message to be transmitted; a computer may initiate the passing of a new message only when it receives that token. Ring networks are not very robust, since a single machine failure will prevent the whole network from functioning.

In a pure physical *Mesh* topology every machine is connected to every other, offering redundancy and fault tolerance; however, this is impractical in reality and hence *Incomplete-mesh* topologies are usually built instead, in which only a selected few of the many possible connections are actually made. The backbone of the internet has an incomplete-mesh topology.

There are several hybrid models that have been created and these are typically *Star bus* and *Star ring*. A star bus uses a bus backbone to link hubs together and run star configurations from those hubs. A star ring uses a hub as a means of transmitting the message or token around a virtual star configuration linked into the hub.

Business value proposition

The networks associated with businesses have evolved considerably since the 1970s, when the majority were based upon mainframe technologies. During this period, mainframes were connected via their intranet to "dumb" terminals, and externally to other companies and remote terminals via modems. During the 1980s, organizations supplemented their mainframes with personal computers that had their own LANs and were also interconnected to the corporate mainframe and other resources. The 1990s saw the mainframe transition from being at the center of the network to being used for specialist high-volume transaction processing requirements, the mainframe being replaced at the center of the network by server architectures. Modern network architectures are typically composed of many types of computing resource, which may include a mainframe, individual workstations and servers, industrial computers (robots), sensor-based technologies (RFID), and remote computing devices (PDAs, laptop computers) connecting via the internet.

In any situation the network topology or design needs to be considered in relation to the goals for that network and, by extension, the strategic goals of the organization. These requirements may be to support a global ERP system, a super-computer performing design work, a LAN of workstations supporting a call center, or for a company to support all of its computing devices on an extended network. IT organizations and network managers use a variety of metrics and benchmarks to determine the performance and effectiveness of their networks, for example end-to-end response time for a user request to be serviced by a web server may be vital to an e-commerce company.

The creation of a network must take into account the volume of data that is to be passed over it, the bandwidth that will be needed, transmission speeds, response time requirements, and the number and type of network devices that will be connected to it. There is a wide array of hardware, software, and systems solutions available and business cases can be developed for varying options, including the total cost of ownership involved over the life of the project, and the scalability of the configuration and the length of time a configuration would take to provide a positive return on investment.

The decision to create a network also involves the construction of a business case in which total cost of ownership is projected upon the basis of a given usage requirements scenario. This may include issues such as the hardware costs associated with each configuration option, the support cost of proprietary versus open-source software, the use of thin client versus fat client, and the use of a value-added network versus the use of the internet for messaging.

Summary of positive issues

Network technologies, including the hardware, software, operating system, and protocols involved, are mature and supported by vendors, contractors, and developers. Networks may be created in a wide variety of configurations from completely homogeneous to completely heterogeneous, and at a wide variety of cost levels depending on requirements.

Summary of potentially negative issues

Network-related technologies are continually evolving and an upgrade strategy needs to be established as part of IT planning. Older technologies and protocols do become deprecated and sometimes eventually unsupported.

References

- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

• W. R. Stevens (1994). *TCP/IP Illustrated* (New York, Addison-Wesley).

Associated terminology: Client–server, LAN, Internet protocol, TCP/IP, ERP.

Network-address translation (NAT)

Foundation concepts: Internet protocol, DHCP, Proxy.
Definition: Using an intermediate proxy server to change the IP addresses in incoming or outgoing transmissions.

Overview

The problem of an organization having too few IP addresses to cover all of its network-enabled devices is only partly solved by DHCP (Dynamic Host Control Protocol, q.v.). DHCP requires that enough IP addresses are available to cover the number of devices actually connected to the internet at any given time, sharing them out on an on-demand basis. As the supply of IP addresses is rapidly diminishing while the number of internet devices is rapidly increasing, DHCP alone is not enough.

There are ranges of IP addresses that have been set aside for special purposes, and are guaranteed never to be allocated as the true IP address of any computer. These are the 16 777 216 IP addresses that begin with "10.," the 1 048 576 that begin with "172.16." to "172.31.," and the 65 536 that begin with "192.168." Any organization is free to use any of that vast number of free addresses, but for internal purposes only. Any data transmitted over the internet using one of those addresses is guaranteed not to be delivered, but an internal network may be configured to handle them in any way that is desired.

Use of these *Private addresses* has a consequence that may be seen as both positive and negative: no computer from outside the private network will be able to communicate with these systems (or vice versa, of course). This is a very negative aspect if the desire is to provide internet access, but it is a very positive aspect from the security point of view. Systems that can't be seen can't be attacked, and employees can not spend office time "surfing the web."

Network address translation (NAT) allows computers on an internal network with private addresses to have controlled access to the outside world. A NAT server must be set up as a member of the internal network, but with a real IP address. The NAT server acts as a bridge between the internal network and the internet as a whole. It accepts all communications intended for the outside world, and changes the associated addressing information, substituting its own real IP address for the real sender's private address, and retransmits those modified communications on the external network. When a reply is eventually received, the reverse procedure is performed, and the response is retransmitted on the internal network to the original sender.

NAT may be used exactly as described above, to allow a whole network of literally millions of computers to access the internet with just one shared real IP address. A similar arrangement allows the NAT server to be pre-programmed to redirect incoming connections to the appropriate private address, so that even publicly accessible servers will be supported.

NAT servers may be configured to monitor network traffic and refuse to forward certain kinds of communications. NAT may be used to prevent all but a select few external sites from being accessed or from having access to the internal network, so it can act as a powerful security tool.

Business value proposition

NAT servers allow organizations to heighten the security of their network by using IP addresses that are not resolvable externally (they are not part of the normal address system used on the internet), ensuring that the organization's computers

can not be found or sent messages through the internet.

NAT servers can also act as security guards, taking a message from a user internal to the organization and then sending it out to an external site, reversing the process should an incoming message be delivered. However, the NAT can be configured so that only authorized addresses can be visited or received from. This would, for example, help to prevent the plans of a new "secret weapon" being sent to anyone other than those working on the project and holding the clearance "Top Secret," if those plans could be recognized in outgoing transmissions. Similarly, because the client computer has no resolvable IP address, it would be much harder for external spies to break into the system.

Summary of positive issues

NAT provides a safe and secure partial mechanism for protecting corporate computers and networks; it can also prevent users from reaching external sites.

Summary of potentially negative issues

Some internet protocols (notably FTP, the file transfer protocol) do not work well through NAT; the consequences of using NAT must be carefully considered.

Reference

- T. Baumgartner and M. Phillips (2004). *Implementation of a Network Address Translation Mechanism over IPv6* (Washington, DC, Storming Media).

Associated terminology: Internet, Host, Network.

Network devices

Foundation concepts: Network, IP (internet protocol), LAN.

Definition: Any device, hardware or software, that can communicate with a network.

Overview

A network device is anything that is connected to a network and able to communicate with it. The best known of these are, of course, individual computers with network adapters (internal network cards or devices added to a free port), but most networks depend upon other kinds of network device: special-purpose hardware with some processing power, but completely devoted to one network-support task. Nearly all of these tasks could be handled by a normal computer, but it is usually more cost-effective and reliable to use task-specific hardware.

A *Hub* is the simplest kind of network interconnection. It connects together two or more network cables and makes them behave logically as though they were a single cable. Every signal that appears on any one cable is repeated exactly and almost simultaneously on all other cables. Hubs provide "signal cleaning," which allows network cables to cover more distance than they normally could without excessive signal degradation, but do nothing to reduce the levels of network traffic that build up when too many devices are connected. Network hubs are almost obsolete, having been replaced by *Switches*, although USB hubs, which are conceptually similar but not a piece of network hardware, are current technology.

A switch is an improved or "smart" hub. Switches also connect together two or more network cables, but apply some intelligence to their processing. A signal that is "heard" by a switch on one of its cables will be repeated on other cables only if it is relevant to the other devices connected to them. Unlike hubs, switches require some setup and occasional management; they must be provided with some data on the structure of the network in order to do their jobs.

Switches are valuable in reducing network traffic and increasing the speed of communication. The amount of traffic on

a network increases rapidly as the number of computers connected increases, until it reaches a point at which the network is saturated and performance degrades sharply. A switch breaks a network into smaller sub-networks with much less cross-traffic.

A *Bridge* is a particular form of switch that usually has just two connections, and is generally used to connect a *Local area network* (LAN) to a larger grouping. Instead of being pre-programmed with information on network structure, a bridge will often "learn" the network for itself by simply keeping note of the kind of traffic that occurs on either side.

A *Router* is similar to a switch, but works at a different level. Switches generally decide whether or not to repeat traffic on the basis of physical identification of the traffic's destination. A router typically looks at the IP address, and may use much more general rules (such as "all IP addresses beginning with '111.222' are found along cable A"). Many devices sold as switches for domestic or office use also perform some router functions.

A *Gateway* is a kind of router that very frequently has the form of a computer running special software, rather than a single-purpose smaller piece of hardware. A gateway is the bottleneck connection between a LAN and the network as a whole.

The terms "server" and "client" properly refer not to an item of hardware, but to a particular implementation style for networked software. A *Server* is a software application that provides some network service; once started, it "listens" for requests coming from other systems, processes them, and sends back responses. Servers are passive systems that wait until some other system specifically requests that they act. A *Client* is a software application that makes use of a server in that manner. Any particular computer may be running a large number of different servers concurrently. Commonly, FTP, HTTP (i.e., web), SMTP (email-sending), and IMAP (email-receiving) services are all provided together. Similarly, a computer may be both a server and a client (for different applications) at the same time. When the term "server" is used to refer to an actual computer, it means either a computer that runs some server software, or a computer that is supposedly more powerful than the average workstation and therefore would be more suitable for running heavily used server software.

Business value proposition

New network devices continue to be developed and planned: these include routers that can "understand" the type of traffic that they direct and facilitate faster, more efficient data routing; elevators (lifts) that can transmit their status back to the manufacturer, allowing proactive maintenance planning; and refrigerators with bar-code readers that can order groceries over the internet from preferred vendors.

The development of network devices that communicate over wireless networks also continues to evolve, involving devices embedded in cellular telephones, automobiles, PDAs, and laptop computers.

Summary of positive issues

There is a very large set of options for network developers to build their networks from, connecting "intelligent" devices, such as PDAs and cellular phones, together through routers, switches, and gateways.

Summary of potentially negative issues

As network device use grows, bandwidth demands will continue to grow and technology upgrades will continue to be required.

Reference

- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

Neural network

Foundation concepts: Artificial intelligence, Cell computing.

Definition: Artificial devices, namely electronic or software simulations, that copy the function of brain cells, connected together in large numbers in the hope of duplicating the function of a brain.

Overview

The original goal of artificial intelligence, and still one of the future hopes for general computing research, is to construct an artificial device, a computer or robot, that behaves intelligently. That is, something that can think for itself, making decisions, having original ideas, and perhaps even having feelings and personality (although the last two might not be so desirable). There were great hopes in the early days of computing, but they were rapidly dashed when the problem was discovered to be much more than just a matter of having enough computing power.

One proposed solution seemed to be foolproof: find out how human brains work, and make machines that do exactly the same thing. If we have an electronic version of a human brain, how could it possibly fail to have the same intelligence? It is now known in great detail just what a brain cell (or *Neuron*) does; they are extremely simple in operation, and it is very easy to implement very exact copies both electronically and in software simulations. A *Neural network* or *Neural net* is exactly that: artificial brain cells connected together as they might be in a real brain. Unfortunately, the human brain contains approximately 100 000 000 000 neurons, each with an average of 1000 connections (or *Synapses*) to other neurons. It is the enormous number of neurons and synapses, and the dynamic nature of synapses, that produces intelligence. It will be a long time before human technology is capable of building anything on that scale.

This view of neural networks as a path to artificial intelligence is not universally accepted. A powerful counter-argument is that duplicating Nature hardly ever produces a viable technological solution. In the early days of flight, many inventors produced machines that attempted to fly by flapping their wings, on the grounds that that is what birds do, and birds fly, so it must work. It never did; aeroplanes do not flap their wings, and nor do they have feathers or beaks. Lawn-mowers do not attempt to emulate goats. Road vehicles successfully use wheels, but nobody has yet made a successful legged vehicle. The argument is that of course successfully duplicating a natural brain would produce something that has all the qualities of a natural brain, but that does not mean that it will be an effective solution or an achievable goal.

In addition, there is the undeniable observation that new-born babies do not make very good workers. The natural brain takes many, many years of intensive training to produce a usefully intelligent being. A successful duplication of the human brain will obviously have exactly the same requirements, plus, of course, exactly the same side effects: personality, desires, and fallibility. How can anyone expect to avoid all of those human traits whilst still developing the same intelligence? This raises the further question of why we would need to create artificial intelligent beings: when the world's population is increasing by about 1 000 000 every day, why put all that effort into making more artificial ones? The typical answer, that artificially intelligent systems could be very useful in environments where humans do not wish to be sent or where the environment is hazardous such as in one-way deep-space flights, becomes cruel and inhumane if intelligence can not be isolated from the other aspects of being human.

Clearly there is a need to create systems that can behave intelligently but do not

need to display cognitive behavior, such as a biometric security system which needs to fulfill only a few clearly statable goals and need not replicate the full behavior or abilities of a human security guard. On reducing the scale of the goals, neural networks can be used successfully, and are an effective solution to some kinds of problems. Instead of having many billions of neurons to create a whole brain, just a moderate number is used, to perform just one small task. With a reasonable number of artificial neurons connected together, some receiving stimulation from input sensors, some sending, having their states read by output indicators, and all widely interconnected, they can, through a process of training, learn to recognize complex patterns in data. This process requires no programming in the traditional sense; neural networks are trained by repeatedly showing them sample inputs together with correct solutions. The trainer does not even need to have any knowledge of how correct solutions can be produced from the inputs, since, if there is a pattern of the right sort, a neural network will gradually learn it, and self-configure to produce repeatable results and interpolate for input patterns never presented in the training data.

These limited scope neural nets, although nowhere near to being an artificial brain, are capable of learning, and do produce useful results in some situations. Neural net based systems have been used in scoring credit applications, optical character and handwriting recognition, and many areas of data mining. A neural net can never *explain* its results; inputs that do not exactly match any that appeared during training may be correctly interpolated, but may also produce totally unexpected *glitches*: inexplicable wrong answers.

Rarely, the term "neural network" is used to describe any network of very simple processors cooperating to perform one task, almost as a synonym for "cell computing." This is not standard usage.

Business value proposition

The much hyped use of neural networks as a mechanism for achieving true artificial intelligence has clearly not succeeded. However, the nature of the techniques makes them suitable for many smaller, well-defined application domains. Applications have been developed and deployed in the areas of healthcare, finance, and biometrics, amongst many others.

In healthcare, neural network technology has been used to help physicians in a variety of areas, including clinical diagnosis and image and signal analysis, as well as emergent areas such as drug development. Specific applications include assisting in the screening tests for cervical cancer, the detection of acute myocardial infarction, the prediction of metastases in breast cancer patients, the prediction of coronary-artery disease, and testing for cirrhosis of the liver.

Diverse neural network applications have been created in many business areas, including stock market analysis, futures and commodities trading, risk analysis, and predicting bond price movements. However, many of these systems are proprietary to their developers and open details of their methods are thus scarce. Other business applications include the use of neural networks for detection of credit card fraud, and for focused direct marketing campaigns.

The application of neural network systems to security problems has included systems being developed for facial, voice, and signature recognition.

Summary of positive issues

Neural network technology has a well-developed theoretical basis. The technology has been applied to systems in a variety of domains, including healthcare, business, and security systems. The technology is supported by a variety of tools, platforms, and software from research labs, universities, and commercial vendors. The use of tools

greatly simplifies the development of applications and the training of neural nets.

Summary of potentially negative issues

Neural network technologies are applicable to small specific problems rather than to systems within large ill-defined problem areas. The training of a neural network can take a considerable amount of resources and time. A complete understanding of neural network technologies requires training in mathematics and computer science. The use of neural networks requires that the user understand that the system is not perfect and dependence upon the results implies an understanding of the risk level associated with that system; neural networks do not have the ability to explain their behavior.

References

- J. Freeman and D. Skapura (1992). *Neural Networks* (New York, Addison-Wesley).
- J. A. Anderson (1995). *Introduction to Neural Networks* (Cambridge, MA, MIT Press).

Associated terminology: Machine learning.

Normalization

Foundation concept: Database.

Definition: The conversion or reformatting of data into a standard, rationalized, uniform representation that still provides the same information.

Overview

Fractions all have multiple representations: 1/2, 2/4, 5/10, and 48/96 are all really the same thing. In general use, this does not present a problem, and sometimes has a benefit: the notion of "five out of ten" may be more accurately expressive in some circumstances. However, when a comparison is required, having many representations for the same thing does cause trouble. It is easy to see that 3/7 and 3/7 are the same, but noticing that 27/63 and 3/7 are equal requires some intellectual effort. Normalization provides a single standard representation for all possible values. If all data is represented by its normal form, checking for equality is as simple as noticing that two things look the same.

In computing, it can save a significant amount of processing time if all data is stored in a normal form. No loss of expressivity is incurred, since output filters may be applied to ensure that data is printed in whatever form is desired.

For data items more complex than fractions, the design of a normal form and the procedures for converting data into it can be a complex task, but the benefits are correspondingly greater. The most important commercial aspect is in the design of *Multi-table relational databases*. As an example, consider a small business that accepts orders from a number of regular customers. It would be perfectly sensible for them to arrange their database in (at least) two tables. One table would record full information on all of their customers (name, address, tax identification, account balance, etc.), and another table would record all the orders (customer name, item ordered, date, quantity, etc.). Although the division into multiple tables is a sound decision, the given design of those two tables is not.

What happens when a customer changes their name, an option commonly taken by people and corporations? Naturally the database will need to be updated. Changing the customer table is a simple operation, but the order table will also need to be changed to keep it consistent. Every single record of an order made by the customer will need to be updated, and that is a long procedure. While the update is in progress, the whole database will be in an inconsistent state and unsafe for use. The design is also unsafe; if the customer name for an order is mistyped, and that error

goes undetected, there will be no way for an automatic database system to associate that order with the originating customer.

A set of rules introduced by E. F. Codd in 1972, known as Boyce–Codd Normal Form (BCNF), provides a normal form for relational database design. If the tables of a database satisfy the BCNF rules, then the problems indicated above, and many others, will not occur and the database will generally make more efficient use of disk space, provide faster response times, and be more robust.

Normal forms exist for many other domains. *Disjunctive normal form* is used for logical formulæ; it allows equivalence to be determined by simple visual inspection, common operations to be performed quickly, and direct conversion into hardware designs. *Backus normal form* is used to specify the syntax of programming languages, and permits automatic parser generation. Church's normal form for *Lambda-expressions* is an essential tool of theoretical computer science.

Business value proposition

Normalization enables technologists to develop database systems that are efficient and effective in their structure. This not only facilitates the development, maintenance, and growth of the database as needs require but also allows these activities to be carried out with precision. The high technical standards associated with normalized systems also help to ensure that the systems work to specification. This may require more effort on the part of the technology team, but ensures that the system will not have unexpected problems later and require troubleshooting, which is an expensive and unpredictable activity.

Summary of positive issues

The rules for database normalization have been developed and used for over 30 years and they provide a formal basis for database development. In other areas, normalization of data may be expected to increase efficiency and reliability.

Summary of potentially negative issues

Normalization can require very specialized expertise or an increased training effort on the part of the organization and technologist.

Reference

- E. Codd (1972). "Further normalization of the data base relational model," in *Data Base Systems*, ed. R. Rustin (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: UML, Data-flow diagrams, Entity-relationship diagram.

Object-oriented

Foundation concepts: Programming language, Algorithm.

Definition: A programming style that focuses on data objects and the operations performed on them as a whole, rather than concentrating purely on executable code and treating data as something that passively flows through it.

Overview

Object-oriented is one of the most popular bandwagons for computing professionals and organizations to jump upon. In reality it is a simple logical evolution in programming methodologies that has gradually been solidifying since the 1960s. It is certainly a Good Thing, but not necessarily the shiny bright new technology it is usually thought to be.

In the older, non-object-oriented style of programming, programmers concentrate on designing programs and algorithms that will process data. The data is viewed as a passive entity that flows through the program, and is worked on by it. Starting in a small way with Cobol in 1961, and fully taking off with the far more rational Algol-68 in 1968, programming languages provided ways of encapsulating large and complex data items, even whole databases, as single manipulable objects in a program. Programming was still viewed as producing instructions that say what to do with the data, but the data could be handled as a single well-defined object, not as an amorphous collection of binary digits.

These developments gave rise to the idea of *Abstract data types* (ADTs) as a tool used in the early stages of software development. With ADTs, all of the different kinds of data that a program will have to deal with are fully and formally defined, and an exact specification is provided for all of the valid operations on those data types. Parts of programs are written to implement those operations, and then the whole program may be defined in terms of those higher-level operations, without the programmer ever having to consider the details of the data again. This resulted in much more reliable program development methods, and is the foundation of *Formal methods*. There was great resistance to the adoption of this programming style in the established software industry, with ADTs being derided as a mere academic toy, and claims that "real programmers use Fortran," which does not support ADTs. However, it did eventually catch on. Object-oriented programming is nothing more than the practical embodiment of ADTs.

Object-oriented programming was first presented to the world as an integrated usable product in 1972, with the language *Smalltalk*, designed by Alan Kay and Adele Goldberg of Xerox. It became a commercially usable technique with the introduction of C++, which took the object-oriented style to heart, although it still treated it as completely optional. *Java* was the first major programming language to make object orientation a compulsory part of the design process, and that is undoubtedly a major factor in its success.

The notion of object orientation as a simple clarifying design technique has been largely lost on the world, mainly because C++ added so much additional baggage to it. Some of those additions are undeniably useful, some less so. C++ is the definition of object-oriented programming in many programmers' eyes, so object-oriented programming is often seen as a big, complex, and arcane system, which it does not need to be.

The usual extensions to object orientation can be seen as major improvements in their own right, but they do have a strong complicating tendency that is not strictly necessary. *Inheritance* is the technique of programming that allows a whole new data type to be defined by simply specifying how it varies from an existing data type. This increases code reusability, and in the long term, reliability. *Polymorphism* and *Virtual*

methods are dependent upon inheritance, and make it simpler to define data objects that contain within them diverse kinds of other data objects. This also increases code reusability and reduces testing and maintenance turn-around times. Unfortunately, inheritance and polymorphism, although simple concepts, bring with them a host of attendant concerns that seriously complicate programming languages: what happens when a new data type is defined in terms of two incompatible older data types? What happens when inherited code operates on data types that had not been defined when it was written? There are many other very technical questions that have to be resolved for any system that incorporates these ideas.

Java solved many of the arcane questions by simply forbidding the conditions that lead to them being relevant. Adherents of C++ claimed that this made the language far inferior to their favorite, but that claim has not been supported by the evidence of years of practice. Object-oriented programming can be a very simple technique, and, when it is, it is of great benefit to all concerned: programmers, their managers, and their customers. It is part of the logical progression of software technology, in the same direction as *Structured programming*, which also met with great resistance from industry for a very long time, but is now a *sine qua non* of software design.

Business value proposition

Object-oriented programming was intended to be a methodology through which programmers could produce readable, reliable, and reusable code, and was popularized by the C++ programming language. However, while the theory upon which it was based is sound, the practical use of the programming style has been clouded to varying degrees by misunderstandings and unbroken old programming habits. A primary problem is that many see object-oriented programming and the language C++ as merely different names for the same thing, and C++ is a language of almost impenetrable complexity. Object-oriented programming can be clean, clear, simple, and understandable, as the original object-oriented language, Smalltalk, showed.

The advent of the Java programming language from Sun Microsystems in the mid 1990s gave programmers a somewhat more manageable environment in which to develop their applications, since it prohibited many of the unsafe constructs freely available in C++. The availability and use of Java has helped to reduce the complexity associated with the object approach and has encouraged programmers to produce systems in a manner more closely associated with their original intent, that of facilitating correct, clearly structured, reusable software.

Summary of positive issues

Object-oriented programming allows for the development of readable, reliable and reusable code, if it is used properly. The object-oriented style has an extensive literature and is supported by many major software vendors. Java is becoming more popular as a language and supports the object-oriented style of programming.

Summary of potentially negative issues

The use of C++ and the poor use of Java have in many cases caused complicated, impenetrable programs to be written, especially in those instances when programmers are allowed to believe that complication is acceptable.

References

- M. Weisfeld (2003). *The Object Oriented Thought Process* (Indianapolis, IN, Sams).
- J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991). *Object*

Oriented Modeling and Design (Englewood Cliffs, NJ, Prentice-Hall).
- A. Eliëns (1995). *Principles of Object Oriented Software Development* (New York, Addison-Wesley).
- A. Goldberg and D. Robson (1989). *Smalltalk-80*, 1st edn. (Boston, MA, Addison-Wesley Professional).

Associated terminology: C++, Java, Programming language, Algorithm.

OLAP (online analytical processing)

Foundation concept: Database.
Definition: Online analytical processing is a set of analytical tools that enable users to examine the relationships that exist within a structured data set, usually held in the form of a data warehouse.

Overview

OLAP is a collection of analysis techniques that are applied to data sets in order to determine relationships between the elements of that set. A well-known example is that a supermarket used OLAP to look for correlations in sales, and it identified that customers coming into their stores in the evenings just to purchase diapers typically also purchased beer at the same time, leaving with two items rather than one item. Knowledge of this unusual combination of products purchased allowed the store owners to achieve better stock modeling and demand forecasts.

Typically OLAP is used to manipulate and examine financial or sales data models (e.g., looking at sales at a variety of geographical levels: sales by country, region, district, or zone) and uses statistical and other tools to identify trends and anomalies. To help in the analysis, mathematical and statistical tools are built into the OLAP packages and accessed through a variety of graphical user interfaces that allow data to be presented as graphs, pie charts, linearity charts, and so on.

There are differing degrees of sophistication available in the OLAP applications and the mechanisms through which the data is stored. The term OLAP was originally used by Dr. Ted Codd in 1993 and he used his relational data model (he also created the term *Relational database*) as the basis of OLAP. This is still used in some instances and is known as *ROLAP*. Subsequently to the relational model, the use of multidimensional databases evolved and the terms *Data cube*, *Star model*, and *Snowflake model* were coined to describe the data structures underlying the OLAP implementation. The OLAP model has been extended to be web-based or web-enabled (termed *WOLAP*) and this allows users to access the OLAP via a network connection. A fourth level of OLAP is the use of spreadsheets, which, through their in-built data-manipulation tool sets (e.g., pivot tables) and their graphical tools, can be useful for handling smaller data sets held in relational and non-relational structures.

Business value proposition

OLAP systems are beneficial to corporations wishing to analyze their data sets. The historical accumulation of corporate data provides a potentially rich source of information for companies. The data is typically divided into live transactional data and historical data, namely data that has been extracted from the live data set, transformed into a new data structure, and then loaded into a separate database (which can be a data warehouse). The OLAP systems then manipulate the data to allow for the identification of trends and examination of subsets of data, and present the data in a variety of formats. OLAP systems have several advantages, including the ability to add to revenues through better analysis of sales and marketing, improved customer satisfaction through enhanced product quality and service, provision of better, more accurate reporting, and the

provision of more timely information resulting in faster decision-making across the organization.

The decision to purchase an OLAP system must be made both through the business case and with consideration of the technical issues surrounding the technology choices. Issues such as the scale of the data set that the OLAP can manipulate and the timescales for the manipulations based upon the underlying platform and database server must be considered. The OLAP's compatibility with the database upon which it operates needs to be taken into account, as must the nature of the GUIs through which the data is viewed. The ability to access the data remotely through the internet also needs to be considered, together with the security issues surrounding any remote access of corporate data warehouses.

Summary of positive issues

OLAPs provide tool sets for the manipulation of data sets. OLAPs are available for a variety of database structures. Various vendors provide OLAPs, including specialty OLAP tool sets, OLAPs that are integrated as an aspect of an ERP, and spreadsheet OLAPs for smaller data sets.

Summary of potentially negative issues

OLAPs require correct "clean" data in order to provide high-quality outputs. The establishment of an OLAP and its maintenance typically requires a high degree of technical knowledge and specific training.

References

- E. Thomsen (2002). *OLAP Solutions: Building Multidimensional Information Systems* (New York, John Wiley and Sons).
- N. Pendse (2004). "Drilling into OLAP benefits," *DM Review Magazine*, March.

Associated terminology: Data warehouse, ERP, ETL.

One-way hash

Foundation concepts: Encryption, Security.
Definition: Irreversible encryption, reducing a message to a small "digest" from which the original can not be recovered.

Overview

The idea of irreversible encryption at first sight seems foolish. What is the point of encrypting something in such a way that it can not be decrypted under any circumstances, even with the key and unlimited computing resources? If the data were so secret that it must never be seen again, it could much more easily have been secreted by simply erasing it.

The true purpose of a one-way hash function is to provide a kind of *Fingerprint* or *Signature* for a digital document: some code that can positively identify a document or data set without being able to reveal it.

Take for example the vital message "HELLO." Assign each letter a number in a uniform way (perhaps the simple scheme $A = 1$, $B = 2$, $C = 3$, etc.), so the message becomes 8–5–12–12–15. Then add 1 to the first number, 2 to the second, 3 to the third, and so on throughout the message, producing 9–7–15–16–20. Then multiply all the numbers together: $9 \times 7 \times 15 \times 16 \times 20 = 302\,400$. Finally, take just the middle two digits of the result, giving the number 24. This is not a secure one-way hash by any means, but it is sufficient to illustrate the point.

The number 24 provides a digest of the message. The same message will always produce the same number, but essential information was discarded, so it is not possible to work back from the number 24 to discover what the original message was. There are only 100 different possible results from this oversimplified operation, but there are certainly more than 100 possible original messages, so reversing the process is a logical impossibility.

Just as important is the observation that, if the original message were changed, the final result would change too. With only two digits, there are only 100 possible results, and it would not take long to find a second message that also results in 24. Real one-way-hash functions produce much larger results (MD5 is a prime example, with 30-digit results, and SHA produces 48-digit results), so finding a second message that produces the same result as any given first message is practically impossible.

That is the power of a one-way hash. If you are not concerned with keeping data secret, but simply want to be sure that it has not been modified, simply apply a one-way hash function to it, and keep the result safe (but not necessarily secret). Every time the data is accessed, simply recalculate the one-way hash and make sure that the value is the same. When used in conjunction with a *Public-key* (q.v.) encryption system, one-way-hash functions provide a secure *Digital signature* mechanism.

Business value proposition

One-way hashing is a technique by which companies or individuals may "fingerprint" their documents or data to ensure authenticity and prevent changes. The method is used by some software vendors to allow their customers to ensure that software applications they receive are the original authorized versions and no third party has tampered with them en route, perhaps by adding some spyware to them.

Summary of positive issues

One-way hash functions are inexpensive to implement and easy to use, and many algorithms are openly available. Strong algorithms provide an almost unbreakable mechanism for ensuring the authenticity of a document. The algorithms are widely published and relatively easy to implement.

Summary of potentially negative issues

There is a concern that many of the algorithms that utilize only a small number of bits may have been broken, which would mean that it is possible to produce a fake document that produces the same hash value as the genuine one. In 2004 it was reported that some popular algorithms including MD4, MD5, and SHA-0 had been broken, but the stronger algorithm SHA-1 remains safe, although it is to be phased out from government use by 2010.

References

- B. Schneier (1996). *Applied Cryptography* (New York, John Wiley and Sons).
- NTIS brief comments on recent cryptanalytic attacks on secure hashing functions and the continued secuity provided by SHA-1 (http://crsc.nist.gov/hash_standards_comments.pdf).
- NTIS (2002). *Secure Hash Standard, 2002, August 1*, Federal Information Processing Standards Publication 180–2 (Washington, DC, NTIS) (http://csrc.nist.gov/publications/fips/fips180_2/fips180_2.pdf).

Online communities

Foundation concept: World Wide Web.
Definition: A group of entities that utilizes the internet technologies to interact to effect a common goal, practice, or interest.

Overview

The term *Online community* relates to the concept of an online environment that is composed of members who have a common goal or interest. Examples of online communities can be found in many areas: *Communities of interest*, e.g., a financial web site through which participants interact and discuss investments; *Communities of relationship*, e.g., a forum for cancer-related issues; *Communities of fantasy*, e.g., communities

interested in games such as Dungeons and Dragons or Multi-User Dungeons; and *Communities of transaction*, e.g., online business-to-consumer (B2C) and business-to-business (B2B) communities such as Covisint, a procurement exchange focused upon the automotive industry.

Online communities use a variety of levels of technology to execute their interactions. These include "list servers" that distribute emails; chat rooms; web sites through which emails and messages are posted; electronic procurement software systems through which members buy, sell, and interact; and highly secure, highly reliable data-synchronization exchanges through which members distribute the data pertaining to their products and pricing. The control over access to communities can be unregulated (as with many communities of interest, for which activity and access are self-regulated by the membership), open but regulated by a moderator, or fully regulated by a controlling body.

Business value proposition

An online community can be a powerful binding force that can be harnessed to drive commerce. Commercial ventures such as eBay and Amazon started out in the form of online "communities" within which members regulated and monitored trades and recommended books to each other. List servers can be sponsored or augmented by the provision of advertising and product placement. Electronic procurement systems provide a focused industry- or product-specific location for participants to concentrate their efforts, reducing the need to build their own individual systems.

Summary of positive issues

There is a variety of mechanisms for providing a central location through which participants can interact. The range of technologies allows low-cost communities to be created quickly and spontaneously (e.g., for a specific event such as a marathon running race, or an event such as a planning meeting). At the other end of the technology spectrum are communities built around large-scale technology platforms that enable data synchronization, exchange, and procurement between the members.

Summary of potentially negative issues

Unregulated communities can lead to poor and unreliable data and information. The overhead required to monitor and maintain communities can be high. The costs associated with data synchronization exchanges and communities can also be high. Competitive pressures to join procurement portals can force companies to join communities and be subject to their rules and norms.

Reference

- R. Plant (2004). "Online communities," *Technology & Society*, No. 26.

Associated terminology: e-Commerce/ e-Business

Open source software

Definition: Open source software is any software whose source code is freely available, and is typically not subject to fees or royalties.

Overview

Since the origins of computing there has always been a culture of "free software" available within the computing community. The free software has included software from corporations and individual programmers who have placed tools, games, applications, and "fixes" (patches) on their systems for users to download via FTP or the internet. The Free Software Foundation (FSF) has been a strong advocate of free software since it was established in 1985; its members are "dedicated to promoting computer users' rights to use, study, copy, modify, and redistribute computer programs."

The FSF "promotes the development and use of free software" (http://www.fsf.org/) and is particularly well known for *GNU*, its collection of Unix utilities. Additionally, the Open Source Initiative is an organization that provides more structure to the provision of "free" code.

Open source code is typified by Linux, a variant of the Unix operating system that was created mostly by Linus Torvalds, who worked on the system in the early 1990s and finalized the first version in 1994. He made this freely available via the internet to other programmers, who subsequently added and contributed to the development of the system, which they made into a POSIX-compliant system (POSIX is a set of requirements for a standardized core of Unix). Linux has subsequently been distributed under the GNU General Public License from the Free Software Foundation. This allows its source code to be freely distributed and made available to the general public subject to certain conditions (see http://www.linux.org/info/gnu.html).

Open source became a well-known term as a consequence of the browser wars that occurred at the height of the internet revolution in the late 1990s. The battle of the web browsers gained a high profile during this period because it is a concern central to all internet users and developers. The battle culminated in 1998 when the Netscape Communications Corporation decided to make the client-side source code of their well-known eponymous system available via the internet for free.

Following the Linux and Netscape initiatives the open source community became more formalized under the auspices of the Open Source Initiative (OSI) (http://www.opensource.org/), which was founded in 1998 as a non-profit corporation "dedicated to managing and promoting" what they have termed the "Open Source Definition," which is itself derived from the Debian "Free Software Guidelines." This definition states that, for software to be considered "open source" and recognized as such (they offer certification signified through a trademarked "OSI Certified" logo) the software must satisfy ten criteria:

(1) Free redistribution,
(2) Source-code availability,
(3) Modification to and subsequent redistribution of the code must be permitted,
(4) Integrity of the author's source code,
(5) No discrimination against any persons or groups,
(6) No discrimination against any fields of endeavor,
(7) Distribution of license,
(8) License must not be specific to a product,
(9) License must not restrict other software, and
(10) License must be technology-neutral.

Interestingly, in the early days of computing, a large proportion of all software was free. Software would generally only work on one kind of computer, and, given the extraordinarily high cost of computers themselves, manufacturers would usually throw in the software as part of the package, not having to worry about their competitors' customers being able to benefit from it. Failure to provide and maintain an operating system at no extra charge would have made computers virtually unsellable. This all changed with the advent of cheap personal computers at a price that could not possibly support adequate software development, especially when desktop computers became relatively uniform, so one manufacturer's free software would equally benefit their competitors.

Business value proposition

Open source systems allow organizations to have access to free software for which the source code is available. The support and quality of the code can vary considerably from program to program and

organizations need to be aware of the risks associated with using such systems. However, it is also the case that some commercial vendors, rather than funding the continued support of an aging software package for a small (but sometimes dedicated) group of users, will place the source code in the public domain. This allows users and interested parties to work on the code, upgrade it, and maintain it. This has the effect of keeping the product's users happy (they no longer have to pay a license fee) and having the software supported for free by a user group, and as a consequence the vendor's brand is not weakened by having disenchanted ex-customers.

The open-source movement has developed and maintains products in a very wide range of product areas, including operating systems, word processing, spreadsheets, databases, and internet browsers. This has allowed some companies to use exclusively free-source software in their selection of products and thus is highly favored in organizations and countries that can not afford proprietary vendor-managed software.

Summary of positive issues

The Open Source Initiative attempts to bring professionalism and standardization to the domain of open-source systems. Open-source software is of low cost and frequently under continuous development by user groups. Open source allows users to read, modify, and develop the source code of an application.

Summary of potentially negative issues

Open-source code can vary in quality and might not ever have been rigorously tested. The code might not have any documentation or specifications associated with it. Companies using open-source code need to ensure that the code is actually legally open-source code rather than just copied code that has been released under another name.

The existence of free software can put excessive pressure on traditional software companies, which may well have been producing a far superior product, but may be unable to stay in business in the face of zero-cost "competition." There are enough users who value cheapness over quality and enough uninformed corporate buyers who never have to use the free software that they recommend to make this a serious concern in many areas.

It is not totally clear whether the free-software movement is really beneficial overall. There are situations in which reliable software is absolutely critical, and the typical free-software distribution agreement about providing no warranties and accepting no responsibility is inadequate. If skilled programmers are expected to give away the fruits of their labors for free, they will not be able to devote themselves professionally to the project.

References

- Free Software Foundation, 51 Franklin Street, Fifth Floor, Boston, MA 02110, USA.
- http://www.fsf.org/.
- http://www.opensource.org/.

Associated terminology: Linux (see Unix).

Operating system

Foundation concept: Software.
Definition: Essential, permanently running software that keeps a computer operating.

Overview

In the old days, running a computer required a full-time staff. Programs had to be "fed in" manually from decks of cards or magnetic tapes, then the input data had to be made available in a similar way, and everything had to be tended carefully until the final results could be extracted,

and another program could be started in its place.

As more convenient forms of storage, principally disks, became available, the requirement for human operators became an intolerable bottleneck, wasting far too much of the computer's still very expensive time. The solution was to make the computer operate itself. For each program run, a set of operating instructions would be written, exactly stating which program should be run, which files contain the data, what to do if anything goes wrong, and what to do with the results. These instructions were written in a form that could be understood and obeyed directly by the computer, usually called a *Job-control language*. The job-control instructions could even be stored in a disk file, completely automating the whole process, relegating human intervention to disaster recovery and paper loading.

In order for the computer to be able to understand the job-control instructions, some programming is required. Software must read and interpret the instructions, and carry out the actions indicated. This software was the earliest form of operating system. When a computer is first started up, some special action is required to load and start execution of the operating system's software. From then on, it is self-maintaining; the operating system never relinquishes control until the computer is turned off. Of course, all software can (and usually does) have design flaws; an error in the operating system can have a wide range of effects, from minor annoyances to introducing subtle and undetected errors into the results of programs and destroying data. Some kinds of error will simply cause the operating system to stop running, at which point the computer becomes unusable ("freezing up") until it is restarted; this is known as a *System crash*, and is sometimes visible as the infamous "blue screen of death."

Operating systems today often present two alternative faces to users. The traditional interface involves typed commands and responses, and is familiar to many in the form of *DOS* (Disk Operating System) or the "Dos shell." By modern standards, DOS is a very basic job-control language, capable of running only one application at a time, and arranging for its own execution to be continued only if nothing goes wrong, after an application has finished. "Industrial-strength" operating systems usually have a similar typed command interface, known as a *Shell*, *Monitor*, or *Command-line interface*, that is generally preferred by technical users. The now-discontinued *TOPS* and *VMS*, together with all varieties of *Unix*, follow this pattern. The alternate interface is the now universal *Windowing environment*, or *GUI* (graphical user interface), preferred, or, more precisely, demanded, by non-technical users. Usually this windowing environment is just a secondary interface running on top of a traditional primary one, although the popular *Windows* operating systems have abandoned this model and reversed the situation.

Since these early beginnings, the scope, complexity, and error-proneness of operating systems have all grown enormously. Typical operating systems provide a large library of standard program components to simplify the software-implementation process. Multiprogramming allowed more than one application to be runnable concurrently, so that, when one is temporarily unable to continue, perhaps waiting for a slow input device, another could run in its place. Multiprogramming grew into time-sharing, the system under which multiple applications do in fact run concurrently, with control switching between them perhaps 100 times per second. Virtual memory, which allows more applications to run than would actually fit in the computer's memory, is another system that requires

constant operating-system supervision. Network access requires oversight to ensure that incoming messages are delivered to the right application. The ability to interpret the movement and clicks of a mouse and to provide a visual "desktop" requires much more computational effort than is needed with a traditional job-control language.

The full, efficient, and effective use of an operating system requires a significant degree of training. Many science-fiction books and films depict operating systems as intelligent agents (e.g., the HAL-9000 on the spaceship in the film *2001: A Space Odyssey* and the computer on the *Star Trek* spaceship Enterprise). While it is possible to have natural-language voice interfaces, the "cognizant" systems in these films are still a long way off; their study is an ongoing aspect of research into artificial intelligence.

Business value proposition

The selection of an operating system depends upon the use to which it is to be put, the knowledge level and skill of the operator, and the resources available to support that system. There exists a range of operating systems to meet a user's requirements depending upon the configuration of these variables. One aspect of operating system selection that may override the others is the maintenance of consistency across the organization. Hence an organization may choose to select a GUI-based operating system that a majority of the users will find easier to understand than a command-line interface. Versions of that system may be used across a range of platforms supporting a variety of applications. At one end of the spectrum one version may support the corporation's ERP system, while clients may run another version of the operating system, portable laptops a third, and hand-held devices such as PDAs and cellular telephones yet another version. The use of one "family" of operating system across all these devices benefits the organization by potentially reducing costs, through a reduction in maintenance, training, and license expenses.

A second strategy organizations may wish to pursue is the "best-of-breed" strategy, whereby one operating system is selected to work on the server cluster, another on the desktops, and yet others on the laptops, PDAs, and cellular telephones. Alternatively, a low-cost strategy may be appropriate, implying that freeware and open-source systems can be chosen with less regard to their applicability or usability.

A major concern in operating system selection is the level of support associated with a particular system and vendor. The metrics associated with support level include the cost of the license and the associated maintenance levels (if any), the cost associated with service packs (upgrades and fixes), user groups and user community, conferences, technical papers, and the level of support from application vendors.

It is also important to consider the relationship between the operating system and the applications that are to be supported by that system. This is important in the overall design of the systems architecture (the total combination of applications, operating systems, networks, etc.) because not all operating systems can run all applications or database systems. Equally, some applications such as ERP systems may run in a thin client mode, making the operating system on the client almost irrelevant so long as the client can run a browser. A further issue that is important in many purchasing decisions is the lifespan of the operating system, and the nature of any superseding operating system. The nature of the technology upon which an operating system works will be one factor that determines whether the operating system needs to be changed to stay in alignment with the technology in use (e.g., CPU, memory

systems, network requirements, monitor and peripheral devices). A second factor is the amount of support (and cost) that a vendor provides to transition to the next version. A third factor is the ability of new versions of the operating system to be backward-compatible so that all the previous applications will continue to run.

Simpler items of machinery, such as automobiles and washing machines, are often computer-controlled, and have software that controls their functions (e.g., the engine management system, traction control, braking system, etc.). However, that software is not usually considered an operating system, since it is really just a piece of *Firmware*, and does not control the operations of other software. In some cases, onboard computers do interact with each other, and do run a number of different applications depending on changing circumstances, and many manufacturers use a special reduced-size version of a normal computer operating system to act as the basis of the onboard controller.

Summary of positive issues

There are several types of operating system interfaces available, including command-line-style and GUI-style systems. Applications vendors can provide data regarding operating systems requirements for their system. Operating systems are available in a variety of configurations to support a variety of applications and platforms. Many operating systems have both formal and informal support groups.

Summary of potentially negative issues

The types and variety of operating systems have been consolidating since the 1980s, resulting in three distinct primary camps: the Unix-like, the Windows family, and the others. Some operating systems lack sufficient support from vendors, have limited functional options, contain bugs, and have poor security. Some operating systems are not supported at all except through informal user groups and are considered legacy by their original manufacturer. Some operating systems are not backward-compatible and thus do not support the applications that ran on previous versions.

Reference

- A. Silberschatz, P. Galvin, and G. Gagne (2004). *Operating System Concepts* (New York, John Wiley and Sons).

Associated terminology: Unix, Compiler.

Optical character recognition (OCR)

Definition: Processing a digital scanned image of printed or written words, to isolate and identify the individual letters and the words they form.

Overview

It seems like a very simple problem. Given a good digital image of some printed words, work out what they say. The shapes of characters are well known, and printed words provide a great deal of consistency. Naturally, understanding handwriting could be more difficult, but understanding printed characters seems to be a simple matter of seeing which of 26 already known patterns each letter matches.

Optical character recognition (OCR) is in fact an exceptionally difficult task. Even with the enormous commercial gains to be had from being able to process printed documents automatically, no system has yet been perfected. The problems are numerous. When scanned with enough resolution to give sufficient detail, no two printed characters ever look exactly the same. Variations in lighting, texture of paper and ink, slight imperfections in the image, imperfect alignment, and a host of other problems mean that looking for an exact match with predetermined letter shapes will never succeed. Added to that is the problem that different fonts, type sizes, and styles (bold, italic, etc.) provide a potentially

infinite variability to how any letter may look.

There are many commercially available software packages that do a creditable job of OCR, even down to properly setting paragraphs and diagrams in popular word-processors' file formats. None of them is totally reliable, or even comes close to the degree of accuracy that a capable and careful human reader could achieve. For the collection of essential data, for which accuracy is of some importance, it is always necessary to have a human operator compare the original printed document with the results produced by OCR software. This does not by any means mean that OCR is pointless. Human operators can compare the two versions much more quickly than they could possibly type the entire document, so OCR is a great time saver; it simply means that a totally automatic process can not be expected to produce error-free digital transcriptions of printed documents.

The similar problem of handwriting recognition is many times more difficult. Either users must write in a specially designed highly stylized way (such as the strange alphabet familiar to users of the Apple Newton), or the system must be trained: hand-held through a selection of samples to learn a new writer's style. Even then, reliability is far lower than for printed-word OCR.

Business value proposition

The potential for OCR is vast, since the ability to capture electronic versions of whole collections of documents would move organizations closer to the envisioned paperless office. However, the technology has not yet reached the point at which this can be effortlessly and reliably achieved. OCR software is available to scan documents and store the results in a digital form, but the reliability of these systems is less than total, and it is necessary to cross check the electronic document against the paper original by eye to ensure correctness. Predominantly, OCR technology is effective in application areas that use special optical character sets such as those found on checks. Another application of OCR is in computing devices that accept stylized handwritten input, but these systems also require extensive training of users if they are to work effectively.

Summary of positive issues

OCR can enable large quantities of documentation to be entered into a system (with varying degrees of accuracy). OCR systems that use specially designed characters and that are read in special-purpose readers are effective in automating processes (such as mail sorting). Some computing devices allow hand-written input through the use of special styli and screens.

Summary of potentially negative issues

OCR of documents does not provide 100% reliability of duplication when scanning "real-world" documents that were not specifically created to be processed electronically.

Reference

- H. Bunke and P. Wang (1997). *Handbook of Character Recognition and Document Image Analysis* (Singapore, World Scientific Publishing Company).

Associated terminology: Natural language processing, Neural networks, Machine learning.

Optical storage

Foundation concepts: Storage, Disk, Backup.
Definition: Data-storage systems based on reflection and refraction of light, rather than on the traditional magnetic properties of materials.

Overview

Primary storage in computer systems, that known as main memory or RAM, is now

exclusively built from solid-state electronics; data is stored as static electrical charges. Secondary storage, usually known as disk, is almost exclusively built from spinning disks on whose surfaces data is recorded as microscopic magnetic fields. *Optical storage* provides another alternative, in which data is stored as minute deflections or deliberate defects in a reflective or refractive medium (often just a thin layer of shiny aluminum). Data is read from optical media by shining an accurately focused laser light onto it, and measuring how the reflection of that light is affected.

Laser disks, or *Disco-vision*, were introduced in 1969, but were fundamentally analog systems, and could not have been used effectively for digital data storage. The first optical storage devices suitable for computer systems were the now universally familiar *Compact discs* or CDs. Although CDs were originally used exclusively for audio recordings, they store their contents in a completely digital form, and are ideally suited for all kinds of digital data. A CD consists of two layers of transparent plastic with a very thin layer of metal, usually aluminum, sandwiched between them. Microscopic bumps are printed onto the aluminum during manufacture, or may be "burned" in by laser light in a CD recorder. When a CD is played, another laser shines onto the surface, and the microscopic bumps cause detectable changes in its reflection.

CDs have a usual capacity of 650 MB, but that may be slightly increased in some models. Data may be read from a CD at rates of about 10 MB per second, and CD-recording hardware may fill a CD in about 5 minutes at best. Although CDs do provide good long-term data storage, they do not last for ever. Scratches on the plastic surface may often be repaired completely, or even ignored successfully, but deterioration of the aluminum layer can not be repaired. This deterioration is much accelerated by storage in poor conditions, and occurs more quickly in cheaply made CDs: when the glue holding the two layers of plastic tightly together starts to fail, air can get to the aluminum layer, and that is the beginning of the end. If CDs are to be used for archival or backup purposes, the lifetime of the chosen media must be carefully investigated.

DVDs (*Digital video discs* or *Digital versatile discs*) provided a large increase in capacity, and hardware for writing data to DVD blanks is now as widely available as that for CDs. A DVD uses the same basic technology as a CD, in the same way; improvements in technology between the introduction of the two simply allow for more to be done in the same space. DVDs use smaller bumps, read by a shorter-wavelength laser (visible red instead of infrared) from a thinner sheet of aluminum. The standard capacity DVD can store 4.7 GB of data, although more expensive dual-layer versions with a capacity of 8.5 GB are available. Dual-layer is not the same thing as double-sided; double-sided DVDs are not generally used in computers because of the need to turn them over or have two laser pickups.

The DVD market is confused by a profusion of standards. The most basic is DVD-ROM: this is a read-only format; data must be written during manufacture, and can not be modified later. DVD–R and DVD+R are two slightly different write-once formats: data may be written onto a blank disc once; after that it can not be modified. These are the cheapest and most useful for archival backups and software distribution. Most hardware can read both the –R and the +R versions, but some recorders can write onto only one format. Domestic DVD video players are also often capable of playing only videos recorded in their preferred format. DVD–RW and DVD+RW are re-writable formats; the discs may be erased and re-recorded many (but not unlimited) times. The difference between –RW and +RW is the same as the minor incompatibility between −R and +R.

DVD–RAM and DVD+RAM are extensions of the RW idea; instead of having to erase the whole disc before writing new data onto it, single blocks of data may be erased and re-written individually, so a DVD±RAM disc behaves like a normal but slow hard disk. The DL suffix indicates "dual layer"; dual-layer blanks are much more expensive than single-layer ones, and require special hardware for writing. Currently, only DVD–R DL and DVD+R DL are available.

Two further improvements to optical disc technology have recently become available. Both take the same basic technology even further, using a violet laser to detect even smaller, more closely packed bumps in the reflective layer. One is HDDVD (*High-density DVD*) with a standard capacity of 15 GB, but a 30 GB dual-layer version; the other is called *Blu-ray* (because of the laser color), and has a standard capacity of 25 GB, but also dual- and quadruple-layer versions for capacities of 50 GB and 100 GB.

Magneto-optical discs, which were popular for a period, are not very widely seen now. In a magneto-optical system, data is stored as microscopic magnetic fields on the surface of a rotating disk, but a tightly focused laser light is also applied when data is written. This has the dual effect of allowing smaller magnetic fields to be written (thus increasing capacity) and rendering those fields optically detectable. When data is read back, only a beam of polarized light is needed, no magnetic pickups; the light interacts with the magnetic field when it is reflected, and the data is read from the reflected light beam. Magneto-optical discs are much more complex and expensive than traditional magnetic discs, and, once the price/capacity ratio for the latter had fallen significantly, the former fell out of favor.

Holographic storage, in which data is stored throughout a three-dimensional volume instead of merely on a two-dimensional surface, is the subject of some current research. If successful, it could result in a rapid increase in available storage capacities, and greater reliability since it might not require any moving parts. No such products are commercially available, or promised for the near future; holographic memory currently lingers in the world of fiction.

Business value proposition

Optical storage offers organizations and individuals the ability to store and archive data in a semi-permanent medium at relatively low cost. For several decades tapes and "floppy" disks have been used to archive data; however, these media have the same inherent problem in that the data is encapsulated in a magnetic coating that can degrade over time. The same is true for audio and video cassette tapes, which frequently give very-low-quality play-back after a few years. While computer tapes are usually stored in a temperature-controlled environment and used infrequently, they suffer the same inherent limitations (in some systems the tapes need to be replaced as frequently as every ten backup cycles). CDs and DVDs solve this problem by not using magnetic materials that can degrade, but instead their data is burnt onto the disc, although it must not be forgotten that these media do still degrade in their own ways. While CD and DVD technology has been evolving, its major limitation is the amount of data that can be written onto a disk. "Juke-box" technology for CDs has been developed, allowing many CDs to be created without human intervention or delays.

The blu-ray optical technology has the potential to expand the options open to network users and managers since the data storage capacity is much higher than for older optical storage devices, and the speed at which data may be saved is also much increased. Blu-ray technologies have been designed to accommodate the needs of

the high-definition television communities; however, the technologies will inevitably be adapted to computer-related uses.

Summary of positive issues

Optical storage technologies offer individual users an easy way to store relatively high amounts of data on almost non-volatile media. New technologies such as blu-ray have been developed to provide higher capacity storage at higher data transfer rates.

Summary of potentially negative issues

Optical storage technologies have been limited in size, and over a long period of time the medium itself is subject to decay if not well maintained. The variety of acronyms for CD and DVD media is large and confusing and care needs to be taken when selecting a disc type for use. Holographic storage is currently a research area but no products have yet been demonstrated for commercial use.

Reference

- F. Yu and S. Jutamulia (1996). *Optical Storage and Retrieval* (New York, Marcel Dekker).

OSI seven-layer model

Foundation concepts: Network, Protocol.
Definition: A commonly used abstraction of network software architecture.

Overview

Network applications can be very complex pieces of software, because there are so many different levels of control that they need to exert: from correctly controlling the transmission of signals on the network hardware, through routing data over potentially long distances and complex network topologies, ensuring that all signals are received correctly as transmitted, defining the kinds of interactions permitted, all the way to providing a convenient end-user interface. Such complex systems can be implemented reliably only if they are divided into more manageable subsystems that may be developed independently. For most new development, the ISO's OSI (*Open Systems Interconnection*) seven-layer model is used.

1. The "physical layer" is concerned with the kinds of cables and connectors used, and the nature of the electrical (or other) signals transmitted.
2. The "data-link layer" describes the format of data packets sent on the first layer, and communications when a direct link between two systems exists.
3. The "network layer" controls long-distance communication, when a direct link does not exist, and data packets have to take a number of "hops" to arrive.
4. The "transport layer" specifies how data flows between applications, and handles delivery receipts and retransmissions if necessary.
5. The "session layer" specifies the sequences of messages that must be sent and received in order to achieve particular goals, the language of request and response between connected applications.
6. The "presentation layer" describes the format and representation of the individual messages, and any data that they carry.
7. The "application layer" is the layer of interest to users, which provides the purpose behind using the other layers, namely the application that controls communications and provides, uses, or displays the transmitted data.

Developers can expect the first four layers to be provided as part of any working computer system. If a totally new kind of

application is under development, the last three layers must all be created, but, if an interface to an existing kind of system is being developed (perhaps a new email service or a new web browser), then layers 5 and 6 will already exist, and only the final layer is needed.

At each layer of implementation, the developer may rely upon the previous layers already existing and working correctly. Each new layer simply adds a new level of functionality, making full use of the previous layer. For example, IP (the internet protocol) is a layer-3 protocol, it is responsible for delivering data from any computer to any other anywhere on the network. It uses a layer-2 protocol (such as ethernet) which is already capable of delivering data over a local area network (LAN), so all it has to be concerned with is the forwarding required when two computers are not on the same LAN.

Business value proposition

Protocol stacking based on the OSI seven-layer model is of great benefit to network software developers. The clean division of network tasks into a number of layers allows the developer to concentrate on just the one relevant aspect of design, knowing that surrounding layers are isolated and well specified. Within businesses that are not actively involved in technology development there is less need for the MIS staff to be concerned with all the minutiæ of the OSI model, since the majority of the software in use has been designed to be simply "plugged in" to a network, with all issues related to the protocols already resolved.

Summary of positive issues

The OSI seven-layer model provides a robust theoretical model for all systems developers to construct systems around.

Summary of negative issues

The OSI seven-layer model is frequently seen as *the* way the internet works. It is not. The TCP/IP protocol stack happily existed with four levels long before the OSI model was introduced. Insisting on seeing every system as consisting of seven layers can produce a distorted and confusing view.

References

- W. R. Stevens (1994). *TCP/IP Illustrated* (New York, Addison-Wesley).
- International Organization for Standardization (ISO) (1994). *ISO/IEC 7498: Open Systems Interconnection, The Basic Model* (Geneva, ISO).

Associated terminology: Ethernet, Internet protocol, TCP/IP, Client–server.

Outsourcing

Foundation concepts: Business process re-engineering.

Definition: Outsourcing is the use by a company of a third-party provider to perform a function or implement a process on its behalf.

Overview

Outsourcing has been used in business for decades. As organizations began to understand that they did not have to perform all the functions of business themselves, known as being *Vertically integrated*, they moved toward an outsourcing model, contracting with other vendors to supply components and provide services. For example, an automobile manufacturer may choose not to make brakes at all, but to have a company that specializes in brakes produce and supply them under contract.

The history of outsourcing in the context of information technology (IT) has a slightly different past and rationale from those of outsourcing in the manufacturing sector. In the 1960s not all companies could afford computers, and they utilized "computing service bureaus" to undertake their data processing. During the 1970s and 1980s, as computing became more pervasive, companies began to change their views on

software development. Rather than writing all their own programs, it became much easier to purchase packages, in effect outsourcing their development effort.

Outsourcing remained a useful option for many organizations that wished to supplement their own internal IT organization; in particular, the outsourcing of IT training for employees became a popular option. However, a seismic change occurred in 1989 when Kathy Hudson, the CIO of Eastman Kodak, decided to outsource the company's entire IT operation. This strategic decision changed the way that IT outsourcing was viewed by organizations; instead of considering it an expense, they saw the opportunity to "sell" their IT operations and then pay for a service operated by another company according to a contract.

In the twenty-first century, IT outsourcing has evolved significantly from the total outsourcing experienced by Kodak. While very large total outsourcing contracts still occur, companies now undertake selective process sourcing, which, due to advances in technologies, can potentially be performed at any place on the planet.

Business value proposition

The use of process and technology outsourcing has become a popular mechanism by which organizations can achieve a strategic goal. The goal may be to allow the company to focus on strategic IT initiatives, achieve a higher level of operational performance than could be developed in-house, achieve a level of performance at lower cost than could be achieved in-house, complete a short-term specialist project when the setup costs for in-house development would be prohibitive, move IT assets and staff off the balance sheet, or alleviate the effects of staffing shortages.

Summary of positive issues

The ability to source technology skills around the planet has made a wide variety of options available to organizations. While costs are frequently considered to be a primary driver of outsourcing, other issues (such as the ability to provide a call center that is located in a suitable time zone) also drive the decision to locate IT services in distant locations. Strategic outsourcing enables organizations to use IT to achieve a variety of strategic ends, including cost advantages, human capital advantages, and customer-service provision.

Summary of potentially negative issues

Outsourcing can be associated with a variety of positive attributes, but many of them may be positive in the short term and problematic in the long term. Outsourcing may provide a cash bonus from the sale of the IT function, but, if the outsourced functions are not performed as well as expected, it may be very difficult to bring the IT function back in-house (also known as *Insourcing*). While different locations have different costs associated with them, they also frequently have differing levels of security provision and legal regulations to work within, which may prove detrimental should a problem occur.

Reference

- S. Cullen and L. Willcocks (2004). *Intelligent IT Outsourcing* (Burlington, MA, Butterworth-Heinemann).

Associated terminology: Application service provider, Hosting, ISP.

Packet switching and circuit

Foundation concept: Networks.

Overview

The standard model for a voice telephone system is a *circuit switching* network. When two users or end-stations wish to communicate, a direct connection must be made between them, and dedicated to that call for its duration. The connection may be through copper wires, fiber-optic cables, or even radio or microwave transmissions. It may pass though any number of switching stations, staffed by human operators with plug-boards, or electronically operated. It is even possible, using frequency shifting, for many conversations to be carried by the same wire at the same time. The key concept is that a complete connection must be made, then the conversation can happen, and then the connection may be dismantled.

Circuit switching allows very simple telephony equipment built on nineteenth-century technology to operate effectively (the first telephone exchange was working in 1877), but it does not allow very efficient use of the high-cost infrastructure (long-distance cables and transmitters).

The alternative is a *Packet-switching* network, and requires that all communications are in digital form. In a packet-switched network, every end-station has a permanent connection to at least one switching station; each switching station has a number of permanent connections to other switching stations. The connectivity does not change during use: connections are not made before data is transmitted or broken thereafter; there is a path (perhaps a very long path, through dozens of switching stations) from any end-station to any other.

Digital data is split into manageably sized chunks, known as packets, normally a few thousand bits long. A digital label is added to each packet, identifying the contents of the packet, its sequence number (its "chunk number," which is used to ensure that the original data is reassembled in the correct order), and its destination. The labeled packet is transmitted to a switching station. The switching station receives the entire packet, inspects its label, and retransmits the packet to another switching station, which is (at least hoped to be) closer to the ultimate destination. The packet makes its way, hop by hop, across the network, until it reaches a switching station that has a direct link to the ultimate destination, whither it is delivered.

Each station on the network is responsible only for sending a packet across one single link. The individual packets of a transmission may take different paths across the network, and may even arrive out of order. If one link is broken, packets may simply take an alternate route, or be held in storage until the link is repaired. This gives packet-switching networks a robustness not available with circuit switching. The division of data into small packets, and the loss of the requirement that transmission by the originator and reception at the ultimate destination should be simultaneous, allows the most efficient possible use of network bandwidth.

Carefully designed protocols are an absolute requirement, to ensure that packets are delivered to the right destination and reassembled into an exact duplicate of the original data, to ensure that any transmission errors are detectable if not correctable, and to ensure that the original sender is able to infer that data has not been received when there is a larger problem. The best known and probably most used of these is the TCP/IP/ethernet protocol family.

Frame relay is a term often confused with packet switching. It is certainly a similar concept, but by no means a synonym; the confusion results from the fact that

the data packets transmitted by packet-switching networks, especially ethernet and IP packets, are often called *Frames*. Frame relay is just one of the many possible implementations of packet switching; it was originally developed as a simpler alternative to the X.25 standard, to allow WAN (wide-area network) connections to utilize ISDN lines, and is now commonly used over T1 and T3 lines. Frame relay is expected to be gradually replaced by networks using ATM (asynchronous-transfer mode).

Business value proposition

The development of packet switching has enabled a robust mechanism through which data can be sent reliably from one point to another. Packet-switching technology forms the basis of the internet and is composed of a wide variety of standards and protocols such as TCP/IP and ethernet. The open nature of most of the packet-switching standards and the fact that there are organized working groups actively supporting their ongoing development have led to their widespread adoption.

Summary of positive issues

The technology associated with packet-switching networks is reliable and mature. The technology is extremely well supported and almost universally adopted. Packet-switching technologies are used as the basis of the internet and include TCP/IP and ethernet.

Summary of potentially negative issues

The technology continues to evolve and requires organizations and individuals to deploy resources in order to maintain the compatibility of their systems with current industry standards.

Reference

- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

Associated terminology: IP (internet protocol), Voice over IP.

Parallel processing

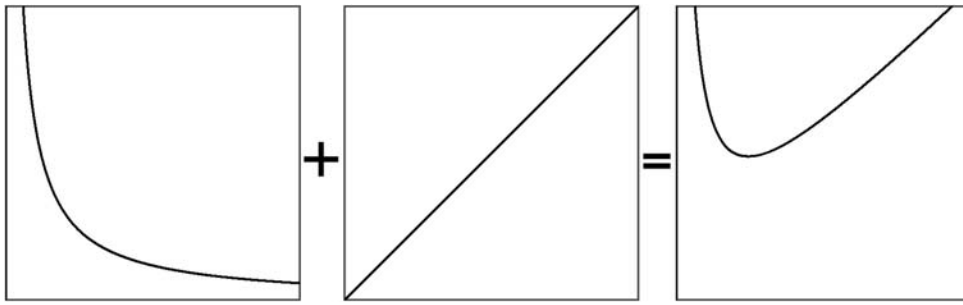
Foundation concept: Algorithm.
Definition: The use of more than one computer or CPU concurrently to solve a single problem or to run a single application.

Overview

It would seem obvious that a number of computers working together on a single problem will produce faster results than a single computer could. That is the basis of parallel processing. As with most things that seem obvious, it is usually not that simple. There are two major considerations that have great influence on the effectiveness of parallel processing.

One is the communications overhead. Most problems can not be devolved into smaller sub-problems that are totally independent. Take for example the elementary-school problem of long division. Two people working together to divide a big number by a smaller one will take just as long as a single person would, because there is no opportunity for them to work concurrently on different parts of the problem. On the other hand, when compiling statistics on a large number of data items, two people can produce a significant speed-up: they can divide the data items into two smaller sets and independently calculate the statistics for their own halves. Even here, they will not be able to double their speed, because some extra calculations will be required to combine the statistics for the two half-piles into overall results. Finally, there are problems for which parallel processing does produce optimal results: two people can eat a cake twice as fast as a single person could; no communications are required at all.

In the general case, the amount of time each processor must spend communicating



with other processors can be expected to grow directly with the number of processors involved. Viewing simple graphs showing (1) how processing time reduces with increasing number of processors sharing the load, (2) how communications time grows, and (3) the sum of processing time and communications time illustrates the point clearly: in cases like this, there is an optimal number of processors to use, beyond which adding more actually produces a slower final result.

The second essential consideration is the algorithmic *Complexity* of the problem at hand. If the algorithm chosen is significantly worse than linear in its time complexity, then parallel processing is very likely to be beneficial. If it is better than linear, the benefits are not so clear, and deeper analysis will be required. For example, consider the problem of sorting a large number of database records into alphabetical order. Naive sorting algorithms, the kinds of method that a person naturally uses, are *quadratic* in time. That means that the time taken to solve a problem grows with the square of the number of data items involved: twice as much data means four times as long to process; ten times as much data means a hundred times as long to process. Now suppose that it would take one person a whole year to sort a collection of one million records into order (that is not a totally unreasonable figure, if the records are small and easy to handle). Two people working on the problem would divide the records up into two halves, with 500 000 records each. Each person would sort their own half, but, because of the quadratic nature of the sorting method used, each person would take only a quarter of a year to sort their share of the records. They would then have to merge the two sorted piles into a single sorted pile, but that operation is an order of magnitude faster than the sorting. So two people could sort the records in just over one quarter of the time taken by one person. Ten people could do it in just over one hundredth of the time. The only reason why 250 000 people working on the problem couldn't do it in a few seconds is the communications overhead: merging 250 000 piles of four sorted records each would take quite some time.

So, there are cases in which parallel processing pays off handsomely, cases in which it has a negative effect, and, of course, everything between those two extremes. Parallel processing is not a simple matter, and needs thorough investigation for any given problem domain. Fortunately, this is a well-researched area, and there is a vast literature available to those who know what to look for.

There are also choices to be made regarding the style of parallel processing to be used. Should there be a number of independent computers connected by a network, or a single specially designed computer with multiple CPUs? The former, known as *Distributed computing*, or a *Loosely coupled system*, allows systems to be built from off-the-shelf components, but has far slower communications; it is the kind of

system that is often used when a problem is devolved into a number of quite large sub-problems, such as with queries on very large databases. The latter, known as a *Tightly coupled system*, is a much more powerful strategy, but very expensive. Very few vendors offer large multi-CPU computers; it may even require customized computer design.

Another possibility is to have a single powerful computer act like a larger number of smaller computers by dividing up its time amongst a number of separate tasks, rapidly switching its attention from one to another, maintaining the illusion that all are being processed concurrently. This is called *Multi-processing*, and is a technique used by all general-purpose operating systems to support multiple applications running at the same time. *Multi-threading* is an emerging technique that allows a single CPU to run more than one program at the same time without having to switch attention rapidly from one process to another. This is related to the concept of a *Thread*, which is a stream of execution within a single application, allowing one application to work on multiple problems without resorting to complex programming techniques.

Business value proposition

Parallel processing is an area of computing that holds great potential for a variety of large and complex problems. It is unlikely that a simple office application such as word processing would benefit substantially, if at all, from implementation on a parallel-processing architecture at current levels of technology, but significant improvements are likely for computationally intensive tasks in the foreseeable future.

The use of a distributed, loosely coupled form of parallel processing is attractive for those with significant computational problems. Many corporate PCs spend as many as 17 hours a day doing nothing (24 hours a day on weekends), and could be assigned to some useful network-coordinated background task in their idle periods. For example, the *SETI@home* (Search for Extraterrestrial Intelligence at Home) research group is looking for a signal from outer space, and relies on vast numbers of volunteers via the internet to load a program that acts like a screensaver running in the background only when the computer is idle, to perform signal processing. It is estimated that they have many millions of users on their network, contributing many teraflops of computing power and trillions of floating-point operations a second to solve the problem. In 1999, a similar cooperative effort involving about 100 000 personal computers worldwide succeeded in cracking a secret message encrypted with 56-bit DES in less than 24 hours. The combined idle time of a large corporation's network of computers is a formidable resource waiting to be tapped.

The use of tightly coupled systems is currently limited to either very simple small systems or highly complex problems requiring extensive processing-time and data-throughput capabilities. One such system is IBM's massively parallel Blue Gene/L computer, which has been designed to scale to 65 536 dual-processor nodes with a peak performance of 360 teraflops (a teraflop is 1 000 000 000 000 arithmetical operations per second). This system is being used to investigate, amongst other things, the problems of protein folding in the biological sciences.

The adoption of loosely coupled and widely distributed processing in the style of SETI@home requires special software to be deployed, and it is clearly not suitable for applications in which security or response time is a major overriding concern, but its applicability to problems in the public domain and for the public good is clear. The adoption of tightly coupled processing is constrained to a certain class of major problems and is also constrained from a user's perspective by the limited number of

super-computers that are available and the expense in utilizing their capacity.

Summary of positive issues

Parallel processing greatly improves the speed of processing for certain classes of problems. Several approaches to parallel processing are available, including loosely and tightly coupled processing. The use of the internet and volunteered computing resources can allow enormous computing power to be unleashed. Very large, tightly coupled parallel processing systems are available for complex problems. There is a vast literature on parallel processing.

Summary of potentially negative issues

Parallel processing is a complex computing environment. Tightly coupled, massively parallel computers are very expensive, difficult to use, and limited in availability. Parallel processing has limited application to the majority of commercial programming problems under current technology.

References

- B. Wilkinson and M. Allen (2004). *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers* (Englewood Cliffs, NJ, Prentice-Hall).
- I. Foster (1995). *Designing and Building Parallel Programs* (New York, Addison-Wesley).
- A. Gara M.A. Blumrich, D. Chen *et al.*, "Overview of the Blue Gene/L system architecture," *IBM Journal of Research and Development* **49**, 195–213.

Associated terminology: Algorithm, Complexity.

Password

Foundation concept: Security.
Definition: A secret identification code used to verify identity.

Overview

The idea of a password is very simple: some word, phrase, or jumble of letters and digits known only to one person, which they use to verify their identity when accessing a secure computer system. The security problems associated with passwords are less well known.

The first problem is with transmitting the password. The user must type their secret password and send it to the secure computer system so that that system may perform the identity check itself. If the user is connected through a modem over a telephone line (dial-up connection), there could be a "tap" anywhere on the line, allowing a third party to see everything that is transmitted in either direction, and easily record passwords. If a network connection is used (ethernet, fiber-optic, DSL, cable modem, etc.), a similar device called a "packet sniffer" could also be attached at any point, and could intercept all transmissions (to be precise, a packet sniffer is a software application that may be run on any computer; it is that computer that taps into the network connection).

The second problem is with password management on the secure computer itself. In order to check that the password entered by the user is correct, the secure system must have some kind of record of every user's password. If security at that installation is not perfect, if they fall victim to any form of "hacking," all passwords could be revealed. If a password gives access to any kind of financial control (such as ordering transfers of funds), how could any user prove that they did not authorize a particular transfer if their password has been compromised?

The third problem is with the choice of password. Many people are required to remember a large number of passwords for the various systems they use, and have to pick something memorable, for obvious reasons. Some users are truly careless and choose passwords like "password," "abc,"

or their own names. The reason this is a problem is that guessable passwords really can be guessed. Someone trying to break into a system can simply try a lot of common password choices for a large number of accounts, and can expect occasional successes. "Cracking kits" are readily and freely available for download: these are software packages that automatically scan through whole dictionaries, trying out an endless stream of possible passwords at very high speed. No real word or simple pattern is safe. Systems that require their users to pick four-digit PINs (personal identification numbers) as passwords could almost be considered pre-compromised.

There are solutions for each of these three problem areas. The first is a software solution called SSL (secure-sockets layer). If both computer systems have SSL installed and elect to use it, every byte transmitted is automatically encrypted using a strong crypto-system. Security is enhanced by the fact that a new, very long encryption key is automatically generated for each session, so long-term eavesdropping provides no additional leverage. Incredibly, it is possible for two systems to securely choose a new encryption key so that they both know what it is, but even an interloper who had complete access to everything transmitted (even knowing both parties' public and private keys) could not know it (the best known of these methods is known as *Diffie–Hellman*). SSL is available as a standard part of nearly all modern operating systems, and costs nothing to use beyond a small requirement for extra computation. It simply requires that the server is set up to use it. When a web browser warns that information is being sent insecurely, it usually means that the server is not using SSL.

The second problem can not be solved perfectly, but may be significantly reduced by simple standard software methods. It used to be thought enough to store passwords in a specially protected file that could not be accessed by normal users. The computer operating system's own file-protection services were used to ensure that only the system administrator and the password-verifying software could access the password file. In modern times, when new security flaws seem to be discovered every day, this is clearly not sufficient. One successful virus, worm, or Trojan-horse attack could reveal everything. The solution is that passwords are not stored at all. When a user sets or changes their password, it is encrypted using a *One-way hash* (a kind of encryption scheme that can not be reversed: it is impossible to recover the original message from the encrypted version, even if the key is known). Only the encrypted version is stored. Each time the user logs in, the password they enter is also encrypted in the same way, and the result is compared with the previously stored version. This means that the password file could be published in a newspaper without any passwords being revealed. Until recently, it was standard for all Unix systems to keep their password files in a universally readable location.

Unfortunately, increased computing power has made that system less than perfect. Since the one-way-hash encryption system is well known, it is possible for a programmer who has access to an encrypted password to create a program that simply tries out all possible passwords, encrypting them all, and waiting for a match. For systems that allow passwords to be of any length, it is not physically possible to try out *all possible* passwords because there will be an infinite number of them, but most users, if left to their own devices, will pick a simple memorable password of five or six letters, and that is an open invitation to automatic cracking software. Cracking kits commonly run through all words in a very large dictionary of many languages, try all short combinations of letters even if they don't make words, and even try adding digits in various places.

Cracking software can be left running in the background for many days until it happens upon a match.

The only way to make passwords at all safe is to use a combination of one-way encryption, plus putting the password file in an inaccessible place, plus keeping every possible security measure in place. Even then, it is essential to prevent users from picking easily guessable passwords (any software system can be constructed to automatically reject passwords that are not long enough, or that appear in some standard dictionary, or even ones that don't contain a sufficiently complex mixture of non-letters). However, if password memorization is rendered too difficult, then users must resort to writing down their passwords in supposedly secret places, and all hopes of security are then lost. If systems designers let users make a totally free choice of passwords (subject to their being long and complicated enough of course), then users will be able to use the same password for each system, and if only one password has to be remembered, it can be quite long and complex. Systems that reject passwords for seeming offensive or politically incorrect provide no benefit (nobody but the user in question ever sees the offensive password), and serve to reduce overall security.

Another effective solution is to deliberately slow down the password verification process. It should normally be possible to verify a password with less than a microsecond of computing, but if the verification process includes deliberate delays, so that it takes a whole second or two, then the cracking kits that work by repeatedly attempting to log in with different passwords will be slowed down to such an extent that they could never scan through all the possibilities. Genuine users will not be disadvantaged, because logging in happens only once per session. This solution does not help in situations in which the encrypted password file has been released, because the cracking software does not then need to go through the official log-in procedure for each attempt.

Another good security measure is to require that the password verification procedure makes a note of each log-in attempt that fails because of an incorrect password. If each user who successfully logs in is informed of the number of recent failures, they will remember if they did not mistype their own password, and at least know that a break-in attempt has been made. It is also common for systems to disable accounts temporarily, making them inaccessible even with the correct password, if there have been some (typically three) attempts at access with incorrect passwords within a moderate period (often ten minutes to an hour). This reduces the number of attempts that an automatic cracking system can make to such a degree that it can not be expected to succeed at all.

Business value proposition

The processes surrounding the assignment and management of passwords in an organization are primarily the responsibility of the chief security officer; in smaller organizations this task usually falls to the network administrator. The approaches taken to secure systems access must be based upon best-practice principles from the assignment of passwords to the storage of passwords as one-way hashes. It must be remembered that the value of the information stored by the system and password "protected" is proportional to the amount of effort that someone dedicated enough will undertake to break the password and access the system. Even though one-way-hash algorithms are typically unbreakable, reports since 2004 have shown that small bit-length algorithms are capable of being broken and thus, if the data is valuable enough, even more secure hash algorithms need to be used.

It is therefore imperative that all organizations and individuals follow the security procedures completely and revise those procedures on a regular basis. It is advantageous to have an independent security consultant examine the procedures and report directly to a senior member of the organization, such as the CIO. This avoids problems such as network administrators providing a log-in account under a false name so that, in the event that they get fired, they can return to the company's network and perform illicit activities. The security scan can cover all access points, remote-log-in practices, encryption methods to prevent the use of packet sniffers, the possibility of keyboard-monitoring programs that capture key-strokes including passwords, and spyware technologies that monitor users to capture password activity. Even the possibility of rogue hidden cameras in buildings being used to capture pictures of keyboard entries needs to be considered.

Summary of positive issues

By following best-practice principles, password security may be incorporated into an overall technology-security plan. Technologies exist to help ensure that passwords selected conform to a pattern that is not simple to break. Most operating systems may be configured to ensure that passwords are changed on a regular basis and that multiple log-in attempts are not allowed, thus reducing the risk of robotic programs running through a dictionary. Large-bit-length one-way-hash algorithms may be used to store passwords centrally in a secure manner. Passwords may be multiple levels deep and strengthened by the use of a physical access device such as a fingerprint or a security swipe card.

Summary of potentially negative issues

No system that uses passwords alone can be considered totally secure and it is necessary to use supplemental technologies and processes to ensure higher levels of security. Security policies need to be established and independently verified. Policies need to reflect the real security risks that a violation may cause. Many users choose the same password for all of the systems that they access. While this may simplify their lives, it does increase the potential harm if ever that password is compromised. However, that is unlikely to be as dangerous as forcing users to write down their passwords because they have too many to remember.

Reference

- R. Smith (2001). *Authentication: From Passwords to Public Keys* (New York, Addison-Wesley).

Associated terminology: Encryption, Hacking, Cracking.

Patent

Definition: A patent is a legally enforceable property right granted to an inventor, allowing the inventor for a limited time to prevent or regulate others making, using, offering for sale, or selling the invention. This right is assigned in exchange for public disclosure of the invention when the patent is granted.

Overview

The establishment of a patent requires a government agency to acknowledge the validity of the patent and assign the rights of that patent to an individual or other legal entity. Most countries have their own patent system and intellectual-property (IP) laws and a large agency to administer them. The US Patent and Trademark Office, the European Patent Office, the UK Patent Office, the Japanese Patent Office, and the State Intellectual Property Office of the People's Republic of China are among the most influential. In addition there is The World Intellectual Property Office (WIPO) which was formed in 1883 to enforce the Paris Convention, a treaty that allows patents

granted in one nation to be enforced in others. The WIPO covers IP rights in the areas of inventions, trademarks, and industrial designs.

The various bodies administering IP rights each have different categories and rules according to which a patent can be granted. For example, in the United States a patent can be granted as the result of a successful application in three categories (Utility Patents, Design Patents, and Plant Patents), whereas the UK has just a single category. The various patent offices also have different rules regarding what is eligible for patenting; for example, in the UK an invention is not patentable if it is a discovery, a scientific theory or mathematical method, an esthetic creation (literary, dramatic, or artistic work), or a scheme or method for performing a mental act, playing a game, or doing business; and the presentation of information or a computer program is not patentable.

The rules for patentability do change from time to time. It is only relatively recently that software methods became patentable in the United States, and it is widely believed that that change was not entirely intentional. While the USPTO has stated that software as a general class is not patentable, it adopted its "Final Computer Related Examination Guidelines" in 1996 in order to clarify what aspect of software is patentable. The guidelines help patent examiners to determine the relationship between the software and associated "processes" or "machines," which are patentable classes. Patents in the United States and the UK allow the patent owner (the "proprietor") 20 years of protection.

References

- http://www.wipo.int.
- http://www.uspto.gov.
- http://www.european-patent-office.org.
- http://www.patent.gov.uk/.
- http://www.jpo.go.jp/.
- http://www.sipo.gov.cn.

Peer to peer

Foundation concepts: Network, Internet.
Definition: The connection of two computing devices or applications of the same status to communicate over a network.

Overview

Peer to peer describes communications between two computers or applications over a network, in which both have the same status. Neither acts as a *server*, in the sense of client–server connections passively waiting for and servicing requests from a client.

Peer-to-peer communications are generally more complex than client–server connections. It is understood that a server must have a known, relatively static IP address in order to be accessed, whereas clients can be very mobile. In peer-to-peer communications, both correspondents are expected to behave as clients. The modality of two peers finding each other, so that communications may commence, is one of the major problems in any peer-to-peer system.

Peer-to-peer communication does have its advantages, not the least of which is security. In a normal client–server system, all communications, whether they be emails, interactive "chats," or exchanged files, go through a server, and remain on that server for some period. There is a risk that a third party may also be able to access private information before or after the transfer is complete, and, when total privacy is required, it is possible and undesirable that the server will retain logs of which clients communicated, where, and when. Peer-to-peer communications (beside using the internet as a common carrier) are direct, and data does not rest on any server. The risk of a server keeping logs, or failing to erase sensitive messages after transfer, does not exist if there is no server.

The public perception of peer-to-peer systems may be unreasonably negative

because of press interest in file-sharing activities that violate commercial copyright, such as dissemination of pirated music and videos. There is nothing inherently underhand about a peer-to-peer system; it is the positive aspect of enhanced privacy that makes these systems (the best known of which are undoubtedly Napster and Kazaa) attractive to those engaged in piracy. Traditional client–server systems, particularly FTP and HTTP, would provide a much more efficient backbone for copying music and videos, but the chances of being detected are much greater, just as they are for legitimate private communications.

Business value proposition

Peer-to-peer computing became a high-profile technology when Napster developed a method for file sharing across the internet. Napster worked by maintaining a master list of resources (files) and the host computer (location) of those resources, which enabled users to easily locate the resources they wanted. A user could then form a direct peer-to-peer connection with the host computer and download the file. Since the resources being swapped were typically copyrighted music files, a US court judged the activity illegal and ordered the Napster index to be shut down.

The Napster model was superseded by a slightly different model typified by Gnutella. The Gnutella system does not have a central server or contain a master resource index, but rather a user makes a request to another Gnutella user for a file. If the second user has the file, this is acknowledged and a peer-to-peer link is created such that the file can be downloaded. If the second user does not have the file, it then propagates the request to all of the other Gnutella systems that it is aware of. This cascading effect continues until the request times itself out or the file is found.

Both of these models can usefully be used for legal file sharing between groups of users; for example, sharing open-source software, non-secure data or other information. Peer-to-peer technologies are not limited to file sharing, but may also allow users to perform tasks remotely, or to perform complex computations using a large collection of connected computers.

Summary of positive issues

Peer-to-peer systems can assure high degrees of security because there is no third party or server through which the information is routed.

Summary of potentially negative issues

The use of peer-to-peer file-swapping technology is open to abuse and illegal activities. The technology can be complex to develop and maintain, and even more difficult to monitor.

Associated terminology: Client–server, Protocol, Parallel processing.

Person month

Definition: A unit of overall effort used in the determination and description of project-development times.

Overview

A *Person month* (PM), occasionally still referred to as a *Man month*, is the amount of work performed by one person working for one standard working month. A PM is in fact the integral of the number of people working over time: it could be one person working for a month, 152 people all working for one hour, or any other equivalent combination.

The PM is used in project management to estimate and record the effort expected to be required, or actually expended, on a project. Traditional software project-management models, originating with the *COnstructive COst MOdel* (COCOMO) proposed by Barry Boehm in 1981, equate cost with effort. In COCOMO a PM is considered to be 152 hours of working time, which averages

approximately six hours a day and takes into account holidays, average sick days, and weekends.

Business value proposition

A standard work period helps project managers to plan development schedules. The European Union has an upper limit to the number of hours an employee can work per month and this number can also act as a basis for PM calculations.

Summary of positive issues

PMs standardize the process of software development and scheduling.

Summary of potentially negative issues

The use of a standard number in project scheduling and management to represent a group of programmers is dangerous. As with any project development, there is a standard distribution of programmer skills in any project team. The use of a single number that equates to programmer effort also might not take into account the variances across projects and the project teams that work together on different aspects of a development.

It is vitally important to take into account the fact that, when more than one person works on a project, a significant amount of time and effort is expended on communications. Forty people working for one hour can not achieve anything close to the same productivity as one person working for forty hours. This is a fundamental flaw in simplistic use of PMs to represent an amount of effort.

References

- F. Brooks (1995). *The Mythical Man-Month: Essays on Software Engineering* (New York, Addison-Wesley).
- B. Boehm (1981). *Software Engineering Economics* (Englewood Cliffs, NJ, Prentice-Hall).
- B. Boehm (2000). *Software Cost Estimation with COCOMO-II* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Software development.

Phishing

Foundation concept: Security.
Definition: Attempting to trick others into revealing personal and financial information through fraudulent web sites or electronic mail.

Overview

Phishing has become a very widespread problem for users of the internet. A criminal organization or individual sends out emails that are an exact copy of legitimate emails from a well-known company that many random recipients are likely to have accounts with. The emails look exactly like official emails from that company, but direct the recipient to a web site for some vital purpose. Often the email will warn that an account is about to be involuntarily closed, or a large bill is about to be referred to a collection agency, and this is the customer's last opportunity to do something about it.

The web site that the customer is referred to will have an address (URL) that seems to be right for the company, and the content will be carefully set up to duplicate the company's real web sites. Of course, it will be completely controlled by the criminal "phishers." Victims may be made to feel more comfortable by the fact that they have to log in using their pre-established user-name and password, but, of course, the site is not checking your identity, it is simply recording the user-names and passwords that are entered. That might be all that happens; stealing a customer's user-name and password is often all that is required for great financial gains, in which case the

customer will quite probably be redirected to the company's real web site with a message saying that the password entered was incorrect. Otherwise, the illicit site may continue to ask for an ever-widening variety of personal details, account numbers, and anything else the customer can be persuaded to divulge.

An alternate means of attack is to set up web sites with names very similar to those of major corporations, but with slightly misspelled names, and just wait for customers to fall upon those sites accidentally.

The key to safety from phishing attacks is to realize that email is an exceptionally unsafe medium, and that, with standard email clients, it is impossible to know where an email message really came from. The return address and all other information in an email header are exceptionally easy to fake. For this reason no reputable company would expect customers to trust email for anything vital. Legally required notices can not be delivered by email, as there is no possible proof of delivery. Any time an email message asks the recipient to visit a web site, for any reason, it must be viewed with grave suspicion, and remember that visiting a web site can be achieved by as little as simply clicking on a link embedded in the message.

If an email recipient is for some reason tempted to visit a web site as instructed, they should first re-read the email carefully. A vast quantity of phishing emails now come from overseas locations, and the use of language is often poor. Incorrect grammar and punctuation are common give-aways, but, of course, correct usage of language is no evidence of legitimacy. As another check, look at the email "internet headers" (with the "outlook" mail client, select from the menu "view" → "options," and they appear at the bottom of the dialog; other mail readers provide the same information but with a different access sequence). The "internet headers" are not comprehensible to most readers, but they contain a list of the names of some of the email servers involved in transmitting the message. Look through them quickly, just the endings, which will normally show two-letter country codes for any foreign server. If an email that purports to come from a company that you frequently deal with went through a server in an unusual or remote location it is unlikely to be legitimate. For example, if you are in the United States it would be suspicious if the company you deal with to buy fruit in Washington State is routing a message through an Asian or West African site.

Of course, phishing does frequently originate from within one's home country, and the only way to be really safe is very simple. Don't click on links in email messages unless you positively know the sender.

The word "phishing" is just a respelling of "fishing," which is really what it is, fishing for information. The change of "f" to "ph" is just a strange modern fad of no significance. The US Identity Theft and Assumption Deterrence Act of 1998 makes most phishing criminal, but is of little help when the perpetrators are untraceable or overseas.

Business value proposition

As a customer service it is vital that organizations educate and inform their customers about the phishing phenomenon. Organizations involved in internet commerce must at least give the appearance of taking a proactive position. Failing to respond when customers report suspicious emails can destroy customer confidence, and it takes very little effort to invite customers to report suspicions, and in turn warn them of any known form of attack, or to inform them that you would never send an email directing them to follow a link to a web site. Such a stance, and, if possible, taking legal action against the offenders,

helps to preserve a brand and maintain customer confidence in the organization.

Summary of positive issues

There is no positive business value from phishing unless your business is actively criminal.

Summary of potentially negative issues

In individual terms phishing can be extremely damaging for any victim of information, identity, or data theft. Phishing can weaken customers' confidence in an organization that does not have a policy to deal with phishing violations against its customers.

References

- M. Whitman and H. Mattord (2004). *Principles of Information Security* (Boston, MA, Course Technology).
- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Cracking, Hacking, Law cross-reference.

Port

Definitions:

1. Reconstruction of software designed for one kind of computer to work on another.
2. A communications connection point.

Overviews

1: Most software is originally designed to work on one particular kind of computer under one particular kind of operating system. Multi-platform software is a highly commendable target, and certainly is achievable, but it is unusual for software that was designed to operate under one kind of system to work perfectly on another without a significant outlay of additional development effort. It is very common for software to be developed without any special consideration for cross-platform compatibility, then, when it is working and has been well tested, and its value has been assessed, it is *Ported* to another system.

For well-written code, and when the two systems are in some way similar (perhaps two varieties of Unix), the porting process can be almost trivial and does not require any great expertise in programming or even full understanding of the software itself. For applications that make use of system-dependent features, in contrast, porting can be a major project.

2: Every computer has a number of connectors at the back into which other things, such as keyboards, mice, printers, network cables, and USB devices are plugged. These connectors are known as *Ports*.

Network ports, TCP/IP ports, and internet ports: The same word is also used to describe a *Virtual socket*, most commonly for network connections. Even though a computer normally has just a single network port, with a single network cable plugged into it, it can maintain a large number of simultaneous connections with other computers. This is made possible by the operating system, which creates tens of thousands of virtual ports or *Network sockets*.

The underlying internet protocols TCP/IP and UDP/IP allow messages to be given two-part destination addresses. The well-known *IP address* (q.v.) specifies which other computer the message should ultimately be forwarded to; the second part, the *Port number*, is used once the message arrives to decide which of the many concurrently running applications the message should be delivered to. Network-enabled applications obtain a port number from the operating system on start-up, and use that port number as part of their unique address for receiving messages. Although a computer may have up to 65 500 network ports, they are all just virtual constructs accessed through the one physical network port at the back of the computer.

Serial ports: nine or fifteen individual connectors in two parallel rows. Serial ports

are almost obsolete and usually go unused on modern computers. Serial ports were used to connect to external dial-up modems or very old mice. Communications through a serial port occur as a series of single bits in a long stream; since only a single bit can be sent at once, serial port communications are usually quite slow. The speed and other details of bit transmission are not fixed, and must be set on a per-device basis. Some serial devices *Auto sense* the settings, and are almost plug-and-play, whereas others require an exact configuration to be pre-set, which causes trouble when the instructions are inevitably lost.

Parallel ports: 25 individual connectors in two parallel rows. Until recently, parallel ports were the only low-cost means of connecting external devices to a single computer with a fast transfer rate. Parallel ports were used to connect printers and scanners, but have now largely been replaced by USB.

PS/2 ports: small round sockets with six individual connectors and a small rectangular tongue, normally used for connecting a keyboard and mouse, but they are in the process of being replaced by USB. PS/2 ports are usually color-coded, mauve for the keyboard and green for the mouse.

USB ports: small rectangular connectors with a central tongue. USB (Universal Serial Bus) is a relatively new technology that adapts the old idea of serial ports to an age of "smart" devices. A USB device has some minimal processing power of its own, and can accurately identify itself when plugged into a system and recognize commands intended for it rather than for other devices. This means that a moderate number of USB devices may be connected to a single port (through a *Hub*). Improved technology means that USB devices can communicate at speeds that were impossible for serial devices. There are two standards in current use: USB 1.1, which allows communication at 12 000 000 bits per second; and USB 2.0, which allows communications at 40 times that speed. USB 1.1 (also sometimes called Full USB) is now used only on older or low-end equipment. USB 2.0 (also called High-Speed USB) is fully backward-compatible, so any USB 1.1 device may be plugged into a USB 2.0 socket. USB 2.0 is rapidly replacing all of the other ports normally found on the back of a computer (keyboard, mouse, serial, parallel, and SCSI). At such high signal rates, it is essential that good solid connections are made through properly constructed and undamaged cables; the only slight disadvantage of USB 2.0 is that the cables tend to cost a little more.

SCSI ports: SCSI (Small Computer Systems Interface) became an ANSI standard in 1981, and rapidly became the connection of choice for the high-speed connections needed for external disk drives, CD-ROM burners, high-resolution scanners, and other devices. Its original transfer speed of 40 000 000 bits per second was much faster than anything else available for PCs or any other small-to-medium-sized computers. SCSI now has many versions, which vary in their levels of compatibility: Fast, Wide, Ultra, Ultra-Wide, Ultra-2, Ultra-2-Wide, Ultra-3, and Ultra-320. The fastest of these are still faster than USB 2.0 (Ultra-2-Wide can transfer data at 640 000 000 bits per second), but SCSI is far more expensive, always requiring costly shielded multi-core cables, and usually requiring an additional interface card in the computer.

Business value proposition

The term port has two major meanings, as described above. The first refers to transplantation of software from one system to another. This may be highly desirable, and much depends upon the degree to which the original designer utilized functions specific to a particular computer or operating system. If the design is highly specific in its use of a unique hardware resource, then portability will be limited or may require significant effort. If the system uses only standard, non-customized

features then the system's potential for portability is greatly enhanced. The management of a software development team must enact policies requiring the use of standard resources if portability of the final product is desired.

The second use of the term refers to a computer's physical connectors, typically located at the back of desktop computers, into which external devices are plugged (although it is now common for USB connectors to appear at the front of machines due to their frequent use and the difficulty in plugging devices into a small connector slot at the back hidden from view). Physical connections are made for the internet, and for devices such as printers, mice, scanners, and external hard drives. Typically, the trend is toward standardization of connectors, and USB is now very much the favorite. As USB becomes the de facto standard, devices that connect through serial ports will become obsolete as computer manufacturers drop support for this type of device. Virtual ports, such as those used for internet connections, need to be considered as part of the security assessment of the network, since ports connected to insecure applications allow hackers to illegally access a system and cause problems.

Summary of positive issues

Standardization of ports on computers allows connections between pieces of equipment to be made more easily. The use of virtual ports allows a large number of connections to be made by devices without the need for a large number of physical ports.

Summary of potentially negative issues

The move to standardization of ports (USB) will reduce support available for other, older port types such as serial and parallel ports. Virtual network ports need to be secured to prevent intrusion from hackers.

References

- D. Groth (2003). *A+ Complete* (Hoboken, NJ, Sybex–John Wiley and Sons).
- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).
- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Network devices, Cables and connectors, TCP/IP.

Power protection

(UPS, Battery backup, Surge protection, Line conditioning, Power conditioning)

Definition: Devices designed to prevent damage to computers and other electronic equipment caused by or borne upon electrical power-supply lines.

Overview

Since the introduction of solid-state electronics, the intrinsic reliability of the main parts of a computer has become exceptionally high; CPUs fail only in fabricated excuses. Unfortunately, there are still external sources of failure: any component with a moving part will eventually fail. One of the most significant causes of failure is the power supply: even a computer with 100% reliability will not work without electricity.

There are four common kinds of power-supply problem that lead to computer failure, they are as follows.

Power failure. Someone accidentally pulls the plug out, or a circuit breaker trips, or a power line needs to be repaired. For whatever reason, electrical power fails to reach the computer. It is very rare for this to cause any *physical* damage. Whatever operation the computer was performing will be unsuccessful, but on restoration of power a computer is normally expected to remain in good working order. However, damage to data can be extensive. Applications must be written with special care to

ensure that data integrity is maintained during a power failure.

An *Uninterruptible power supply* (UPS) is usually a rechargeable battery unit: it is plugged into the normal mains electricity supply, and the computer is plugged into it. Normally, the UPS remains passive, but, when the mains supply fails, the battery automatically takes over and temporarily provides power to the computer. The changeover is so fast that the computer is totally unaffected. The capacity of any battery is severely limited, so a UPS will keep a computer working for no longer than a few minutes. This is enough for most power failures, and in critical installations should be enough time to start up an emergency generator.

UPS hardware ranges from small low-cost devices that provide a few minutes' protection to a small computer, to industrial-strength units with their own electrical generators that automatically start before the batteries become discharged, and could keep a whole installation running for days without an external power source.

Surges. In areas with very primitive public utility infrastructure, electrical supply lines are often above ground, exposed to the weather. If lightning strikes the distribution line, there can be an enormous surge of power with the potential to destroy any connected electrical equipment. Lesser events (such as transformers switching) can cause lesser, but still damaging, surges. A *Surge protector* is a small unit connected in the same manner as a UPS. The most common design uses a simple semiconductor device, a *Varistor*, which shunts the excess voltage to the electrical ground. The varistors can be destroyed by a severe surge, leaving the computer unprotected from subsequent surges: the indicator light should be checked frequently. A UPS usually incorporates a surge protector.

Under- and over-voltage. It is very difficult for a public utility to maintain exactly the right voltage at all times. Particularly in industrial areas, the turning on and off of large inductive loads (essentially anything with a big electric motor) can result in voltage fluctuations, both over and under the nominal value. Modern computers tolerate a wide range of supply voltages, but there are always limits. A UPS constantly monitors the supply voltage, and should automatically switch to battery power not just for total failures, but also for the duration of excessive voltage swings.

Incorrect wiring. It is not unheard-of for the electrical system in a building to be incorrectly installed: the live and neutral conductors can be crossed over, the ground conductor can be left unconnected, etc. These errors can lead to safety problems themselves, and can prevent other safety equipment from functioning. A surge protector can not divert excess voltage to ground if the ground wire is disconnected. Inexpensive wiring checkers are widely available, when plugged into the electrical supply they indicate via red and green lights whether the wiring is correct.

Electro-magnetic interference (EMI). Whenever wires pass through a varying magnetic field, stray currents are induced in it. All electrical equipment creates varying magnetic fields, so the modern office is something of a farm for induced currents in wires. The introduction of high-frequency signals into electronic equipment can have unfortunate effects, so power-protection equipment often includes EMI filters. Many power cables have EMI filters built in; they are small thick tubes that appear to be clamped to the wire.

Business value proposition

UPS devices are used to prevent unexpected systems outages that can lead to the loss of data, hardware damage, and business downtime. The devices are universally available, in a variety of specifications, and are

easy to install, with relatively little maintenance required. UPS devices allow system administrators to bring a system down in a controlled manner if the duration of a power failure is expected to exceed the UPS's battery life.

Summary of positive issues

UPS technology is understood and well supported by vendors and manufacturers worldwide. The lifespan and quality of units vary, but they can be purchased with 10-year warranties. UPS systems are available in a wide variety of configurations (e.g., dual-voltage outputs). Many manufacturers provide extended-run battery options and "hot swapping": exchanging batteries while the UPS is running a live system. Some vendors provide complimentary software to monitor the system, provide alerts, and shut down or restart equipment automatically, and to assist with maintenance and provide self-test logs.

Summary of potentially negative issues

There are different battery technologies available; older ones such as nickel–cadmium have a "memory effect" that prevents full charging after a partial discharge. They need to be completely discharged in order to regain their full operational capacity. Battery devices can only be a temporary solution to a power outage and must be supplemented with generator capability if longer-term secondary power-supply requirements are needed.

Privacy Act of 1974

Definition: The US Computer Matching and Privacy Act is a "code of fair information practices" that attempts to regulate the collection, maintenance, use, and dissemination of personal information by federal executive branch agencies.

Overview

The Computer Matching and Privacy Act was first enacted in 1974 and provided a legislative framework through which the information and data collected on individuals (US citizens and legal aliens) could be stored and compiled. The act has periodically been amended to keep pace with technology. Its key provisions are to

(1) prevent unauthorized matching of disparate federal databases or the matching of federal databases with non-federal databases;
(2) ensure accountability by maintaining records of who accessed which data and when;
(3) ensure that only pertinent data is maintained on individuals by federal agencies;
(4) require that the name and location of the "store" or database be published in the federal register;
(5) maintain data that is used for making "determinations about individuals";
(6) maintain accurate records, which includes allowing individuals to gain access to their data and amend the data when necessary;
(7) "maintain no record describing how an individual exercises rights guaranteed by the First Amendment unless expressly authorized by statute or by the individual about whom the record is maintained or unless pertinent to and within the scope of an authorized law enforcement activity";
(8) enforce code-of-conduct rules for technology workers maintaining and developing the systems; and
(9) enforce security measures to prevent unauthorized access to information.

These and other aspects of the act are overseen and coordinated through the data-integrity boards that are mandatory in each federal agency.

Business value proposition
The act provides a framework through which US federal agencies must operate. The act also has provision for contractors working with US federal agencies and details their obligations.

Summary of positive issues
The act protects personal information and data at the federal level.

References

- D. Solove and M. Rotenberg (2003). *Information Privacy Law* (New York, Apsen).
- M. Rotenberg (2003). *Privacy Law Sourcebook 2003* (Washington, DC, Electronic Privacy Information Center).
- 5 USC §522a (2000).

Associated terminology: Database, Law cross-reference.

Privacy Protection Act of 1980

Foundation concept: Security.
Definition: The US Privacy Protection Act of 1980 protects the "work in progress" and "sources of information" used by journalists from access by law enforcement prior to public dissemination.

Overview
The US Privacy Protection Act of 1980 ("PPA") (Title 42 United States Code Section 42, 2000aa) enacts a provision based upon the First Amendment of the US Constitution protecting "work in progress" and the "sources of information" used by journalists from access by law enforcement prior to public dissemination. Government officers and employees are forbidden from searching for or seizing work materials that are intended for publication in a newspaper or book or for public broadcast. There are, of course, exceptions when danger to life or limb is imminent, or when the author is believed to be involved in a related crime.

Business value proposition
Registered journalists are not required to divulge sources of data or information pertaining to work in progress to law-enforcement agents, thus allowing them to examine potentially "sensitive" topic areas. Encryption technology may be used to prevent unauthorized access to data and work in progress.

Summary of positive issues
Encryption mechanisms can protect data during all phases of a journalistic investigation. Data storage and data transmission are vulnerable to unauthorized access, e.g., internally, at the internet service provider, at the source, and at the destination. Encryption technologies based upon a wide range of techniques and with various degrees of security are available.

Summary of potentially negative issues
Encryption adds a layer of complexity and overhead to the technology used by the members of the journalistic project. The accidental loss of encryption keys can result in sensitive information being irrecoverably lost.

References

- D. Solove and M. Rotenberg (2003). *Information Privacy Law* (New York, Apsen).
- M. Rotenberg (2003). *Privacy Law Sourcebook 2003* (Washington, DC, Electronic Privacy Information Center).

Associated terminology: Encryption, ISP.

Programming language

Foundation concept: Algorithm.
Definition: A compromise language, accessible to computers and specially trained humans, that allows instructions to be written clearly and unambiguously.

Overview

Computers and human beings do not work in the same ways. Contrary to popular belief, computers are vastly ignorant and not in the least bit intelligent. The only thing they are good at is following instructions, and even then those instructions must be written in exactly the right way. Even when it comes to calculations, the things that everyone knows computers are best at, computers are clueless. A computer would be incapable of working out a simple calculation like $2 \times 3 + 5$ unless someone gave it exact step-by-step instructions.

The level of detail required in the instructions given to computers can be quite mind-numbing. For example, to tell an Intel Pentium how to work out $2 \times 3 + 5$, the instructions are like this: "Put the number 2 into A, then multiply whatever is in A by 3, then add 5 to whatever is in A," and those instructions must be encoded in a special computer-oriented form (called *Machine code*) like "B8 02 00 00 00 6B C0 03 83 C0 05" before they can be obeyed. All that just addresses the simple calculation; it says nothing of the far more complex operations required to display the result.

In the very early days of computing, that is exactly what programming meant. First work out how to solve a problem, then work out how to represent that solution method as a sequence of exceptionally small detailed steps, then encode it into the strange numeric form that the computer expects, then try to get that (now unreadable) program into the computer without making any mistakes. Programming was a very difficult labor-intensive task that only a very few highly trained experts were capable of.

Early on, an inspirational breakthrough occurred, and it was realized that, with an exceptional outlay of effort and ingenuity, it would be possible to write a very complex program that tells the computer how to take something written in a moderately human-friendly form, and convert it automatically into the incomprehensible numeric codes that the computer can obey. It would be as though the computer were programming itself. A human operator could type `"SET A=2; MULTIPLY A 3; ADD A 5; PRINT A"`, and the computer would automatically convert it to "B8 02 00 00 00 6B C0 03 83 C0 05 . . . ," and obey those instructions. This was called *Automatic programming*, and was originally viewed as an aspect of *Artificial intelligence* (AI).

The first program that was able to take a human-friendly set of instructions and convert it automatically into the computer's own internal language was an enormous undertaking, consuming the efforts of the established experts in programming (A. M. Turing, R. A. Brooker, and others) for a great deal of time. The result was the first *Compiler* and the first *Programming language* (known as *Autocode*). The benefits were immediate and obvious. Programmers could spend most of their time on the truly intellectual process of working out how to solve a problem, since writing the solution method in a computer-accessible form took only a fraction of the effort. Also, many errors became easily detectable; if someone enters B9 instead of B8, it could easily go unnoticed, since both are valid, but if someone enters MYLTIPLY instead of MULTIPLY, the error is so obvious that the computer itself could detect it and issue a warning.

Since then, the development of programming languages and their compilers has led to ever more sophisticated systems. A programming language strikes a compromise between the computer's internal numeric codes at one end of the spectrum and straightforward human-oriented or mathematical notation at the other. The continuing development of programming languages generally moves the point of compromise ever closer to the human end of the spectrum, automating more and more of the simpler (or less intelligence-demanding) parts of the process. Modern

languages allow the user to enter an instruction as simple and clear as "Print $2 \times 3 + 5$," and have the computer work out everything else for itself.

Ideally, the development of programming languages would be a linear process, with there being at every stage one language which is the best we are currently capable of creating, and which every programmer uses. There has never been a situation like that. Designers of programming languages have their own personalities and motivations, sometimes take a lot of pride in their creations, and don't always see what is best for their users until it is too late. There has always been a wide variety of languages in use at any one time, with a few (but never just one) recognized as the leaders of the pack. Languages certainly do improve with time, but there is a lot of inertia, since it takes significant effort for a programmer to mentally retool. Even if one programming language were clearly superior to all others, the chance of it being universally adopted would be about the same as the chance of the entire world deciding to speak English exclusively. Furthermore, there will always be the need for future improvements: language development is often driven by new discoveries regarding what *can* be done.

There have been well-meaning attempts to create the universal programming language. Starting in 1958, an international committee defined the International Programming Language *IPL*, which in 1960 became *Algol*. This was a very successful language, and was the basis of a large proportion of university teaching for the next 20 years. As new discoveries were made, Algol gave rise to a whole family of languages, including the very complex but seminal *Algol-68*, and, in a backlash, the very popular but now equally extinct *Pascal*. During the 1980s and 1990s, under the direction of the US Department of Defense, another attempt at a universal language derived from Algol, called *Ada*, was developed; for a while it seemed to be gaining ground, and many government contracts required the use of Ada, but it has now fallen far behind. The current languages *C++* and *Java* are just the latest generation of the Algol family.

Currently, the leading language for professional and technical programming is *C++*, with *Java* a close second. Variants of *Basic* are very popular with non-technical programmers because some programs can be constructed without training in programming, and *Visual Basic* provides simplified interfaces to GUI (*Graphical user interface*) components and common office applications. *JavaScript*, *Python*, and other scripting languages enjoy high popularity due to the ease of performing quick experiments and embedding small programs inside documents.

Legacy languages, particularly *Fortran* and *Cobol*, survive from the late 1950s and early 1960s. They survive because of the large amount of useful code that was written when they were the standard languages; little new development takes place now. Cobol was for a very long time the standard language for business applications, as Fortran was for scientific programming. Folklore still bizarrely insists that Fortran is more efficient for scientific calculations.

For special purposes, various targeted languages exist. Every computer has its own *Assembly language*, which is used for small parts of new operating-system and compiler development. AI research makes significant use of *Logic-programming* languages, such as Prolog, and so-called functional languages like Lisp. There are true functional languages, which have a pure mathematical semantics and are useful in research into programming languages and software engineering.

The standard paradigm for programming is called the *Imperative* or *Procedural* style. In the imperative style, a program tells the computer explicitly what to do as a

sequence of ordered steps. Without special training, programmers tend to think that this is the only way programming could be. An alternative is the *Functional* style, in which a program is a set of pure mathematical functions that specify the correct mapping from inputs to outputs without saying what order the individual steps of the computation must follow. In the pure functional style everything involved in a program is strictly constant, but many languages follow the form of functional programming without following the rules. *Lisp* is an example of this: programs are thought of as functions, but in the mathematical sense they are not, and Lisp does not provide the full benefits of true functional programming. In the logic-programming style, programs are simply logical formulæ specifying the correct relationship between questions and answers, not stating how that relationship is to be realized. The non-procedural styles require great expertise and deep theoretical understanding on the part of a programmer, but, once mastered, they provide major advantages, including rapid prototyping and the possibility of *Proving* a program correct. The *Object-oriented* style is a view of how data and computations should be encapsulated in a program, and may equally be applied to all three of the major paradigms. The two popular languages, C++ and Java, are both imperative and object-oriented.

Business value proposition

Programming languages come in all styles and flavors. They can generally be classified in terms of their ages and "eras." For example, the mainframe era of the 1960s and 1970s was dominated by Algol-type languages and the two major commercial languages Fortran and Cobol. The Algol style facilitated the development of creative programming techniques and led directly to the modern languages Pascal, Modula, BCPL, C, C++, and Java. Fortran became the de facto standard language for scientific programming and Cobol became the de facto standard for commercial systems development. Various attempts were made to produce a language that would be good for everything; PL/I is the most notable, combining features of Algol, Fortran, and Cobol into one big confusing whole. Programming was also performed in a variety of other languages that ranged from assembly language to system-command languages such as JCL (*Job Control Language*), which was used by systems programmers to ensure that programs were executed correctly, and that they had access to the correct resources of the system.

The weight of maintaining and developing the huge amount of Fortran and Cobol code, which was typically poorly designed, structured, and developed, was one of the factors that led to their decline. In the 1980s and 1990s programmers were influenced by the need to develop programs that were more in alignment with the needs of the end user rather than the technologist. They also needed to develop code that was able to operate on a range of systems that included PCs and workstations, and would work correctly on various operating systems. Managerial considerations were also becoming a key influence on which programming languages lived and which died. Project managers were keen to ensure that any code developed was reusable and structured, and could be maintained by any programmer versed in that language. The programming languages also needed to facilitate inter-program communication, since many systems were becoming either embedded inside other systems, or connected as part of a heterogeneous programming environment. This led to the emergence and adoption of several key current languages, chief amongst them C, C++, Java, and Visual Basic.

While the C programming language gives programmers the power to perform assembly-level types of operations within

higher-level programs and to access all aspects of the computer's hardware, it is too easy for inexperienced or untrained programmers to abuse or accidentally misuse the language. This was corrected to a large degree with C++, which retained the power of C but provided a more structured programming environment. C++ has proved to be a popular programming language; however, it is a very complex language to learn and to extract the full potential from. One answer to this was the creation of Java, which offers many of the features of C++ but simplifies the programmer's workload. Visual Basic provides non-technical developers with a mechanism through which they can develop simple programs, and is considered especially useful for creating systems with graphical user interfaces of the type familiar to office and commercial users.

In the 1980s there was a major movement toward developing one standard programming language that would be good for all purposes. The movement was led by the US government, and resulted in the programming language *Ada*. In 1987 The US Department of Defense mandated the use of Ada for nearly all projects under its control, with directive 3405.2 "Computer Programming Language Policy," which stated that "The Ada programming language shall be the single, common, computer programming language for Defense computer resources used in intelligence systems, for the command and control of military forces, or as an integral part of a weapon system." For various reasons, the Ada movement was unsuccessful, and in 1997 the policy was reversed. A new memorandum from the Assistant Secretary of Defense stated "I have directed my staff . . . to eliminate the mandatory requirement for use of the Ada programming language in favor of an engineering approach to selection of the language to be used." As a consequence of this there exists ten years' worth of Ada code in critical systems that are likely to become difficult to maintain, and will become a heavy legacy environment within the US Department of Defense, since very few programmers are being trained in Ada, and it is not an easy language to pick up.

In the 1980s and 1990s there was also a resurgence of interest in AI programming systems and languages, with Lisp and Prolog becoming popular for the creation of knowledge-based systems. The prominence of these systems has declined since then, however, but many of these systems are still in operation. Some KBS programs have been translated into C or used through "shell" environments. Again, these legacy systems pose major maintenance problems for project managers because the support environments that were available at their creation have disappeared or not been updated. Lisp and Prolog require special training, since they do not work in the manner familiar to most programmers.

The use of programming within many organizations to create applications has been on the decline since the 1990s as CIOs began to understand the value of using commercially produced applications such as enterprise resource planning (ERP) systems to replace many home-developed programs. Interestingly, a leading ERP vendor, SAP A.G., chose to write their system in their proprietary language ABAP/4, which not only discourages customization of an SAP ERP system (typically a good thing) but also ensures that SAP A.G. has complete control over the development of the ABAP/4 language and as a consequence their ERP application.

The role of programming languages in the future is likely to continue along a development path that increasingly allows programmers to create systems that are able to interact easily, use the features of the web, provide high-quality graphical interfaces, and operate equally well upon a range of operating systems and hardware platforms.

Summary of positive issues

A very wide range of programming languages is available, including assembler, functional, procedural, logical, and object-oriented programming. Modern programming languages facilitate structured programming, reusability, and portability. There is a huge literature associated with programming, systems design, compiler design, and software project management.

Summary of potentially negative issues

Poorly designed and written programs can cause disaster, and programming correctly requires a significant degree of training. Some programming languages encourage poor programming unless great care is taken in all stages of design and implementation. Incorrect choice of programming language can lead to expensive failures or poorly performing systems. The wide variety of programming languages available can be a negative aspect: *nobody* is expert in, or even knowledgeable about, all of them.

References

- R. Wexelblat (1981). *History of Programming Languages, Volume 1* (New York, Elsevier).
- T. Bergin, R. Gibson, and P. Gordon (1986). *History of Programming Languages, Volume 2* (New York, Addison-Wesley).
- Memo: "Command, Control, Communications and Intelligence," Assistant Secretary of Defense, 6000 Defense, Pentagon, Washington D.C. 20301–6000 April 29, 1997.

Associated terminology: Compiler, C++, Java, Fortran, Cobol, Logic programming.

Protocol

Definition: A set of rules governing the interactions between a number of systems or components.

Overview

Protocols cover a very wide range of situations, from remembering to bow and say "Your Majesty" when speaking to the Queen, through the Kyoto Protocol which seeks to govern how nations interact in controlling and trading carbon dioxide emissions, HTTP, the language of communications between web browsers and web servers, and IEEE 802.3, which specifies how messages are communicated on an ethernet network, to X3.131, which defines the operations on a SCSI bus. A protocol is simply an agreed set of rules that govern how communications are to be performed, no more and no less.

Since modern computing is dominated by communications, protocols abound. If an acronym ends in the letter 'P', there is a good chance that it is a protocol name (FTP, HTTP, ICMP, SMTP, etc.). Some are backed by the authority of major standards organizations, such as the ISO, ANSI, and BSI; some are proprietary systems belonging to individual corporations; some are ad-hoc creations designed by a programmer to meet an immediate need.

When a protocol exists for a given system, it is normally considered to be essential to follow that protocol exactly. Failure to do so usually results in failure of the system. However, it is not unknown for products to be made deliberately to diverge from the governing protocol, in an attempt to undermine a protocol seen as unfavorable to the developer's interests, to add a unique "finger print" to the product so that unauthorized use of intellectual property may be detected, or in an attempt to add "out-of-band" communications undetectable by compliant components. Whatever the reason, the result of variation is often failure.

Business value proposition

The ever-increasing ability for computing devices to connect together, to run

programs, and to enable applications to interact with each other is directly attributable to the adoption of standard protocols by manufacturers and vendors. Protocols have, since the 1980s, moved away from proprietary corporate ownership toward being open and accessible, usually with documentation available for download over the internet.

Many protocols are managed by formal bodies such as ANSI, ISO, and W3C, which foster development through working groups. These groups issue calls for comment from their membership, and steadily develop existing protocols, keeping them up to date.

It is important for all organizations, especially those heavily involved with development of leading-edge technology, to be aware of the protocol formulation process and to play an active part in the working groups that develop them. This ensures that they are not caught by surprise by any new protocols or changes to existing protocols.

Summary of positive issues

The development of open protocols is accelerating and the move toward standardization on commonly used protocols that are non-proprietary will reduce administrative and cost overheads associated with maintaining software compliance.

Summary of potentially negative issues

The fact that protocols change frequently requires that organizations be aware of the changes and their implications. The fact that a protocol exists does not imply that it is good; many poorly designed or out-of-date protocols exist and some are perpetuated for reasons of self-interest by their designers rather than in the true interest of their adopters.

Associated terminology: Internet protocol, TCP/IP, ANSI, ISO.

Proxy

Foundation concept: Network.
Definition: A system on a network that "stands in" for another, performing some or all of its normal tasks.

Overview

In general terms, a *Proxy* is a system that intercepts communications intended for another system and deals with them itself. Proxy servers perform a useful function when a local-area network (LAN) is protected by a *Firewall*. The firewall may be configured to prevent any access to computers inside the LAN from any outside, perhaps as a security measure. When some controlled access is desirable, reconfiguring the firewall might not be an acceptable solution; instead, a proxy server may be used. The firewall may be pre-configured to accept any communications from the designated proxy, and the proxy may be easily configured to accept all communications destined for the protected network, and filter them. Those that are considered innocent are forwarded; others are simply ignored. A strong firewall together with a flexible (but secure) proxy server allows complete control over access to and from the internet.

Another use for proxies is to provide a fast-access *Cache* of frequently requested material. For example, if there are certain web documents that are very frequently needed by users in a LAN, copies of those documents can be kept on a local proxy server. All web accesses can be intercepted by that server, and inspected. If an access is requesting one of the already-held documents, the copy may be sent back immediately in response without requiring any further traffic on the external network. If the requested document is not already held on the proxy, the request would be retransmitted to the real server. The proxy may be configured to monitor the response and keep a copy of the new document so that

future accesses to the same may be handled locally.

This second kind of proxy is also used more controversially to filter internet communications. Proxies may intercept webpage accesses, file transfers, and even email messages, and scan for content that is deemed somehow inappropriate. Applications of this range from corporations attempting to prevent leaks of industrial secrets, or to prevent employees from wasting time with web browsing, to shielding minors from explicit sexual material, or even to preventing customers from seeing rivals' materials.

An even more controversial use of proxies involves modifying material that is sent or received without the author's consent or even knowledge. If a user can access the internet only through a proxy, then that proxy has complete control over all of their communications. One application of this is adding *Banner advertisements* to email messages, but far more sinister uses are easy to imagine.

Business value proposition

Proxy servers provide a mechanism through which a network may be protected from unwanted external messages. They also add value by providing a mechanism for servicing requests without having to place unnecessary extra loads upon the corporate network. A proxy server, whether acting as a filter or providing redirection services, does not need to be physically located at the business, but may instead reside at the ISP telecommunications company's site, reducing the bandwidth requirements on the corporate internet connection. The provision and maintenance of proxy servers may in many cases be sensibly outsourced to a web-hosting company.

While there are costs associated with the setup, maintenance, and operation of proxy servers, the cost is offset by the decrease in internal network traffic that results, the reduction in malicious data traffic entering the corporate network, and the potential for using third-party providers for specialist services.

Summary of positive issues

Proxy servers are strong mechanisms for protecting corporate networks. The technology is mature and widely understood by network managers. Proxy servers may act as a mechanism for servicing web requests quickly without having to route every request internally and thereby increase network traffic. Third-party providers are available to offer specialist services such as spam filtering.

Summary of potentially negative issues

Proxy servers that are used to filter spam and other traffic need to be maintained and monitored to keep them current and effective.

References

- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).
- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Internet protocol, DHCP, Firewall, Network-address translation.

Public key–private key

Foundation concept: Security.

Definition: An encryption system using two keys, one for encryption and a different one for decryption; one key is made public, whereas the other is kept secret.

Overview

Public-key/private-key systems (often just called *Public key*) are based on asymmetric *Encryption* algorithms that require two separate keys (or pass-codes) for each communication. Data that was encrypted using one key can be decrypted only by using the other key, and vice versa; it is not possible

to deduce one key from the other. RSA (named after Rivest, Shamir, and Adleman, the system's inventors) is the best-known and most respected of such systems.

For each individual participating in a public-key/private-key system, one matched key-pair is created. The keys are usually very large numbers, hundreds of digits long, so that a *Brute-force attack*, systematically trying every possible key to decrypt an intercepted message, is impossible. One of those keys is given in absolute secrecy to the individual; it is known as their *Private key*. The private key must never be revealed or left unsecured under any circumstances. The other key, known as their *Public key*, is published as widely as possible, and becomes part of the individual's public digital identity. Public keys must be well known and easily accessible. This arrangement permits a wide variety of secure communications.

Digital signature: if you wish to make a statement in such a way that everyone reading it can have absolute confidence that what they are reading is exactly what you wrote, simply encrypt it using your own private key. Everybody in the world has access to your public key, so they can easily decrypt it and read the message; the fact that your *public* key did successfully decrypt the message is proof that your *private* key must have been used to encrypt it. Only you have your private key, so you must have sent the message. Successful encryption methods are carefully designed so that the tiniest change to an encrypted message will result in absolute gibberish when the message is decrypted, so illicit modifications are impossible.

For your eyes only: if you wish to send a message and be sure that only person X can read it, simply encrypt it using person X's freely available public key. The encrypted message may be sent over a completely open and insecure channel; it doesn't matter who intercepts it, since only person X has the private key required to decrypt it.

Total secrecy: a message from person A to person B may be encrypted twice, once using person A's private key, then again using person B's public key. The result is that only person B can read the message because only they have the required private key, and person B can be confident that it really did come unmodifed from person A.

Private records: naturally, any participant may encrypt their own personal secrets using their own public key, and be secure in the knowledge that only they can decrypt them.

Key exchange: The asymmetric encryption systems needed for public-key systems are more thoroughly trusted, but much more computationally demanding than the usual symmetric systems, so the transmission of large amounts of data can be very difficult. A valuable technique for large secure transmissions is to make up a totally new key for each transmission, and send that key only using the slow but very secure public-key system. Once the recipient has received and decrypted this one-time key (or *Session key*), it is used to encrypt the confidential data under a fast but perhaps less trustworthy symmetric encryption system, and never used again. Many cryptanalytic attacks rely on having a collection of intercepts all encrypted with the same key, so the ability to use a new key for each transmission, and be sure that that key can not be intercepted, provides a significant increase in security.

Business value proposition

The use of public-key/private-key technologies has a wide range of applications. Primary amongst them is the encryption of data to be communicated between two or more parties and of data to be held within a company or by an individual. Emerging uses include the incorporation of encryption into a variety of processes such as voting and election mechanisms, smart cards, digital time stamps, authentication

methods, and watermarking of intellectual property (e.g., music).

Summary of positive issues

A variety of public-key/private-key encryption mechanisms have been openly published and are widely available. These mechanisms enable data to be encoded and decoded with relative ease in an automated manner. The mechanism can also be used to encode data for use within an organization for security properties. The fact that an encryption mechanism has been widely published is a strength, not a weakness: many cryptanalysts, not all of them employed by secret government agencies, will have tried to crack it; keeping success secret for a long time is almost inconceivable.

Summary of potentially negative issues

Many public-key/private-key encryption systems are vulnerable to a cracking technique known as *Chosen plaintext*. If a code breaker is able to design a specific message, and persuade somebody to encrypt it with their private key, and is able to intercept the result (so that they have in their possession both the encrypted and the unencrypted version of a carefully chosen text), it can result in the key being deduced. The simple but essential rule is "Never encrypt a stranger's message." When encrypting for the purposes of digital signature, always make a cosmetic change, perhaps just to the spacing of the document.

References

- B. Schneier (1996). *Applied Cryptography* (New York, John Wiley and Sons).
- K. Schmen (2003). *Cryptography and Public Key Infrastructure on the Internet* (New York, John Wiley and Sons).

Associated terminology: Encryption, Password, One-way hash.

Quantum computing

Foundation Concept: Bit.
Definition: The use of quantum mechanics, the laws governing sub-microscopic particles, in the construction of computing and communication devices.

Overview

Quantum computing is firmly in the domain of science fiction, and can be expected to stay there for quite some while. Very basic experiments confirming that the underlying ideas are at least viable have been performed successfully, but the construction of usable computing or communications devices is at best very distant. We are currently at a stage akin to having discovered that electricity does actually exist; people can start thinking about the possibility of building computers, but the possibility is far off, and it may never happen.

Quantum mechanics governs the behavior of very small particles, single atoms and smaller, and shows that they do not behave as experience of the larger world would lead us to expect. The two key concepts that *might* turn out to be practically useful are *Superposition* and *Entanglement*.

In conventional computing, a switch or logic element is either on or off, 1 or 0. Ten logic elements give ten things that can be on or off in any combination, resulting in $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 1024$ different states. A search through 1024 possibilities is performed by stepping 10 such elements through all possible combinations of states in turn. A single logic element at the quantum scale does not have to be either on or off, it can, by virtue of the principle of superposition, be both on and off at the same time. Ten logic elements can all be both on and off at the same time, which means that in combination they can be in 1024 states at the same time. Thus 1024 different computations could be performed simultaneously with one set of circuitry. Of course, the number 10 was chosen just as an example: with 30 gates, over 1 000 000 000 different states could be achieved simultaneously; the numbers are unlimited. The ability to perform any number of computations in the time currently taken by one computation would have an incredible effect in all areas.

The second feature is easier to picture. Two particles can be entangled, which means that they are always tied together in the same state, regardless of how far apart they are or what objects may separate them. If they are thought of as switches, then turning one of them on instantly results in the other one turning on (and vice versa). This provides a method for transmitting logic signals within a quantum computer, but is also believed by some to provide a possible mechanism for faster-than-light communications: two people vast distances apart could communicate by repeatedly switching the state of a shared pair of entangled particles. It would be impossible for any interloper to "tap into" such communications. This possibility remains highly controversial.

Business value proposition

Quantum computing does not yet exist in any practical form. If it ever does, it will completely change the face of computing, and probably all other human endeavors, in ways we can barely imagine.

Summary of positive issues

Quantum computing could potentially provide an unlimited increase in the power and speed of any computing device, and perhaps even permit instantaneous very-low-power secure communications.

Summary of potentially negative issues

It will not exist for the foreseeable future.

Reference

- M. Hirvensalo (2001). *Quantum Computing* (New York, Springer-Verlag New York, Inc.).

RAID (redundant array of

Foundation concepts: Disk, Storage.
Definition: A system for making a number of disk drives act as a single one, for improved access speed, reliability, or both.

Overview

Of all the critical components of a computer system, the disk drive is by far the most unreliable, and often the worst bottleneck for speed.

A very simple system for gaining almost perfect reliability would be to have two identical disk drives instead of one, and store every piece of data on both drives simultaneously. Naturally, both drives could fail, but the chance of both failing at exactly the same time is beyond remote. As soon as one drive starts to exhibit problems it can be replaced with a new one, primed with a fresh copy of all the data. This system is known as *Mirroring*; it gives almost perfect reliability, and has no other impact on performance. Having three identical disks at all times allows replacements to be made as needed without interrupting system operation.

A relatively simple system for improving access speed can also be implemented with multiple disk drives. One of the limitations on disk speed is the capacity of a single *Track*: only one track of data can be read in a single operation without a delay caused by switching to another track (see *Disk* for further details). Tracks typically contain only 50 000–100 000 bytes of data, so reading a typical file can involve a great many track-switching delays. If the contents of a file are carefully distributed between two disk drives, so that perhaps odd-numbered tracks are stored on one and even-numbered tracks on the other, access speed can be doubled, with data being read from one drive while the other is waiting for a track switch. This simple scheme is known as *Striping*, and may be extended to more than two disk drives for even greater speed-ups.

RAID, a redundant array of independent (or sometimes inexpensive) disks, is a standardized system based on these two ideas. There are seven different forms of RAID, known as *Levels* 0–6, which use different forms of mirroring, striping, and a combination of the two.

RAID level 0 uses only striping, with any number of disk drives configured to behave as a single one. It provides a great improvement in speed, but a loss in reliability, since a failure in any one of the disk drives results in the failure of the whole system, and, with two disk drives, failures are twice as common as with one.

RAID level 1 uses only mirroring, usually with exactly two drives configured to act as one. It behaves exactly as described above, providing exceptional reliability with no cost in terms of performance.

RAID levels 2 and 3 use striping and a reduced form of mirroring. The data is spread across multiple drives, together with a *Digest* of that data (known as *Parity bits*, *Hamming codes*, and *Error-correction codes*). This gives enough information to correct small errors, and reliably detect larger ones.

RAID levels 4–6 use ever more complex systems of striping, mirroring, and error correction, to achieve different balances of performance, reliability, and cost. The original idea of RAID came from research carried out at the University of California in Berkeley in 1988. It quickly became very popular, and is now an industry standard; relatively inexpensive PC motherboards are widely available with built-in hardware support for RAID arrays.

Business value proposition

RAID technology allows data to be stored with a variety of options to improve data storage and recovery. Each RAID level has

its strengths and weaknesses that are determined by the level of fault tolerance, the input and output data rates that can be employed, the reliability levels, the technical performance levels of hardware required, and the cost. A cost-based business case must be made to determine the technology level required for a given organization.

Summary of positive issues

The technology associated with the basic open industry-standard RAID levels is mature and, as disk storage costs decrease, the deployment of larger RAID systems becomes more cost effective in terms of dollars per megabyte. RAID continues to be developed and some companies have created proprietary RAID levels to address specific needs they perceive (or wish to create) in the market. The hardware and software associated with RAID systems are highly reliable and very large data storage is available. Each RAID level has its own positive dimension: RAID 0 is simple and easy to implement; RAID 1 has 100% data redundancy, protecting against a disk failure; RAID 3 and 4 facilitate very high data-transfer rates and have a very efficient mechanism for limited error correction; RAID 5 has the ability to write data at a very high data rate; and RAID 6 provides extremely good error recovery, but incurs a substantial penalty when data is written onto the disks.

Summary of potentially negative issues

Each RAID level has its own limitations: RAID 0 is not actually a "redundant" mechanism since it is not fault-tolerant; failure of any one disk can result in significant data loss. RAID 1 is very inefficient in terms of its use of storage capacity. RAID 5 can be difficult to rebuild in the event of data failure. The higher RAID levels can be very complex to implement, but that is only a concern for a very few developers.

References

- D. Patterson, G. Gibson, and R. Katz (1988). "A case for redundant arrays of inexpensive disks," in *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data*, ed. L. M. Haas and A. Tiwary (Seattle, WA, ACM Press), pp. 109–116.
- D. Vadala (2002). *Managing RAID on Linux* (Sebastopol, CA, O'Reilly Press).

Associated terminology: Backup, Information-lifecycle management.

Rapid application development (RAD)

Foundation concepts: Software development lifecycle.

Definition: Rapid application development is a software development methodology designed to improve the overall speed of system implementation.

Overview

The *Rapid application development* (RAD) approach to software development has its roots in iterative prototyping and computer-aided software engineering (CASE), both of which were popular approaches to software development in the 1980s. The approach was first developed as an iterative software development methodology by Scott Shultz at DuPont in 1984, and was refined and popularized by technology author James Martin in 1991.

The RAD approach is to develop software in pieces, using software tools to speed the development of prototypes. It uses teams typically composed of four to six members that include managers, users, and developers, and the aim is to have a team develop a piece of the total system in 4–6 months. A significant degree of support is afforded to the teams through software tools and development environments. The project methodology underlying RAD is a standard phased model, with an orientation phase, a training phase, business modeling, data

modeling, process modeling, code generation, testing, and, finally, production. Ideally the RAD team members will have undertaken significant prior training and will be ready to go, allowing rapid commencement and execution of a project.

Business value proposition

The RAD methodology is well known amongst developers and versions of it are incorporated into the development approaches used by many consulting organizations. The approach is well supported and tools are available to developers. It relies upon iterative prototyping and this enables the development team to deliver prototype solutions to users more quickly than would be possible using stage-based methods (such as the waterfall method), which present solutions only at the end of the development cycle.

Summary of positive issues

RAD has the potential to deliver software solutions faster than stage-based models do. It allows the development of components in parallel and simplifies the amendment of components should the specification change during development. RAD has significant support in the form of software tools and consultants. RAD works well with other techniques for structured systems development such as *Joint application design* (JAD).

Summary of potentially negative issues

The RAD approach requires that the teams be trained in the RAD techniques and philosophy. The approach works best when the scope of a project is constrained and good use can be made of tools and *Application program interfaces*. Owing to the fragmented nature of the development, with the project development being divided amongst many groups, optimization issues need to be carefully considered if speed constraints are important. The fragmented aspect of the development, with user input to each segment, may result in a loss of overall consistency and deviation from the original specification.

Reference

- J. Kerr and R. Hunter (1994). *Inside RAD* (New York, McGraw-Hill).

Associated terminology: Joint application development, CAD/CAM.

Reliability

Foundation concept: Software development lifecycle.

Overview

People normally think of "one in a million" as being the archetypal statement of long odds, and a failure rate of one in a million describes a very reliable thing. However, a modern computer system can execute 2 000 000 000 operations or more per second, and an error rate of one in a million means that it will go wrong 2000 times every second. That would be totally useless. The reliability required of even cheap toy computer systems is higher even than that required of pacemakers and rocket-ships, not because of any critical need, but simply because the speed of operation magnifies the probability of error.

The reliability of electronic hardware is usually expressed in terms of the *Mean time between failures* (MTBF). If a large number of identical devices were all turned on at the same time, and used continuously until they were all broken, the MTBF would be the average time that each one survived. The arithmetic of failure rates is not necessarily intuitive. For example, consider a system consisting of three components, A, B, and C. A has an MTBF of 1000 hours, B has an MTBF of 1000 hours, and C has an MTBF of 2000 hours. What is the overall MTBF of the system? The solution is to consider a long period of time, such as 100 000 hours. In 100 000 hours, we would expect

component A to fail 100 times, B to fail 100 times, and C to fail 50 times, giving a total of 250 failures in 100 000 hours. This translates back to an MTBF of 400 hours for the whole system.

Unfortunately the situation is not always so straightforward. Component failures are not usually evenly distributed. Consider a simple battery as a counter-example: a new battery put into an electric clock usually works for about a year; perhaps the MTBF for clocks with new batteries is 10 months. If you have 100 new clocks, and put a new battery in each, all at the same time, you will not find the clocks failing at an average of 10 per month. Many months will pass without a single clock failing, then, at around the 10-month mark, they will all fail at times very close together. The MTBF for the system of 100 clocks will be only slightly less than the MTBF for one of its component clocks. It is this non-linearity of the failure distribution that makes systems consisting of literally millions of components (such as computers) able to work at all.

The reliability of software is of a somewhat different nature. Unlike physical components, software that was constructed perfectly does not break after extended use and suddenly fail. No matter how heavily it is used, there is no wear and tear on software. If software ever fails, that means that it was faulty right from the very beginning, and the fault simply had not previously been observed. Expressions of the mean time between failures for software are somewhat deceptive; what they really express is how long the manufacturer expects you to be able to use the software before becoming aware of a fault that is already there. Manufacturers can only guess at how users will use their software, so the mean times are no better than guesses themselves. The only real measure of the reliability of software is the number of faults it already contains. If it has no faults, it is reliable; if it doesn't, it isn't. Sadly, most manufacturers can only guess at the number of faults contained in their software.

The fact that automated space probes manage to reach their destinations in the outer solar system after traveling for many years, and successfully beam back photographs of previously undiscovered moons, is a testament to the fact that people *can* make reliable software if they really try hard enough. One might find oneself asking why software for home and commercial use is so unreliable. Sadly, in most cases the answer is simply that it *can* be. Unlike all other engineering professions, there is absolutely no professional regulation for software engineers; expensive software is unguaranteed, and even comes with user agreements explicitly stating that it might not work and that the manufacturer will not take responsibility. It costs more to produce a good product than a bad one, people continue to purchase software from manufacturers of other software that they already know to be unreliable, and well-trained programmers cost more than untrained ones. Human nature and market forces ensure that general production software will remain unreliable for as long as it is allowed to.

Business value proposition

The ability to create reliable software is a function of the development methodology utilized to create that software. The range of software development styles and methodologies is wide, ranging from the very informal styles in which the system requirements are informally encoded, if at all, in informal language, through to the use of mathematical equations to specify required behavior, a type of development known as *Formal methods*.

The discipline of software development has not yet evolved to the point at which the programmers, analysts, and other technologists involved in the development processes have all become "professional

engineers" in the sense that architects or civil engineers have in their professions. Professional bodies such as the ACM, IEEE, and BCS have developed professional codes and membership levels such as Member or Fellow, and further accreditation levels such as Chartered Engineer and Chartered Scientist have also been designated, but there is no requirement for practitioners to be members of any professional body, nor is there any enforcement of minimum standards of competence.

Programmers and other computer technologists do not need any accreditation, and any responsibility for product quality rests solely on the management team. Managers can demand that everyone in their development team should be members of a professional organization such as the IEEE and have Chartered Engineer status in order to work on a project, and they can further stipulate that the team use a formal methodology such as Z or VDM and that the specifications be refined mathematically until they are encoded in a program. Managers rarely make such demands, since the increase in costs would be significant.

An organization that is procuring software may also specify that the development team and the software vendor as a whole must meet Software Engineering Institute (SEI) Level 5 Engineering certification, the highest level of certification, which requires that code is produced under certifiably high levels of process control. The business value proposition for this style of development is that the systems will be of very high quality and extremely well documented, and that future changes to the system can be made with a complete understanding of the consequences. The problem is again that this style of development is extremely expensive and technologists who can perform at this level are extremely rare. A trade-off is inevitable; organizations would like to have all their systems rigorously developed but probably would not wish to incur the cost and make the effort required. Thus the management issue on software reliability comes down to questions of balancing the total cost of ownership of the software over its life, including liability issues, against return on investment for a project.

Summary of positive issues

The reliability of software and hardware within a computer system can be managed by developing software through the adoption of appropriate levels of formality in the development process using the SEI's developmental guidelines.

Summary of potentially negative issues

Without strong management and process controls in place, the development of software is subject to large degrees of variance in terms of quality, methodological style, and programming technique, leading to a corresponding variability in reliability.

Reference

- D. Ince (1991). *Software Quality and Reliability: Tools and Methods* (London, Chapman and Hall).

Associated terminology: Formal methods, RAID, Y2K problem.

RFID (Radio-frequency identity) tags

Definition: Radio-frequency identity tags are small, relatively low-cost label devices that can be used to automatically identify and track objects.

Overview

The idea of automatically identifying an object from a distance gained prominence and became practical during the Second World War when aircraft carried transponders tuned to a certain radio frequency, and, upon receipt of a particular signal,

they transmitted back an individual identifier to state that the aircraft was friendly. Subsequent to the war the concept of a reflected-signal device was discussed in the writings of Harry Stockman, but the technology available at the time was not adequate to implement his theories.

Since the 1940s many people have worked on remote-identification technologies and the concept has been refined considerably. The first patent was not awarded until 1973, when Mario Cardullo was awarded a US patent for an RFID device (the patent expired in 1990). Technical implementation problems remained for some considerable time and it was not until the late 1980s that the technology was able to be put to practical use as a means of allowing drivers of cars to pay highway tolls without stopping at a toll booth. The tags used in cars are known as active tags because they have their own internal power source (usually a battery) and this enables them to transmit their data further and in a wider field than is possible with so-called passive tags that have no internal power supply and can only reflect or interact with an externally generated signal.

Active tags hold significant advantages over passive tags in that the internal power source enables signal quality to be higher whilst providing a greater range of reliable operation. However, they are also much more expensive to manufacture and may require occasional human intervention, for example to change the batteries.

While active tags are useful, their cost and need for maintenance prohibit them from being used in mass commerce, especially in situations where they would be discarded after a brief period of use. Thus, passive tags have been the subject of intensive research and development, and, as the cost of their production has dropped, more commercial opportunities have been realized. A passive tag system works in a very similar way to the active tag system, but the passive system, having no energy source of its own, needs to convert the incoming radio waves into energy and use that energy to transmit back data.

The amount of data that can be held on passive RFID tags is growing and products with 2000 bytes are available (the Fujitsu MB89R116 has 2048 bytes of total memory, 48 bytes of which are used by systems programs), with a "data-retention" period of 10 years under normal conditions.

There are current moves toward the creation of protocols and standards for RFID tags. Primary amongst these is the effort by EPCglobal Inc., a joint venture of GS1 (formerly the EAN), GS1-US (formerly the UCC), and a group of universities, whose focus is to "establish and support the EPCglobal Network as the global standard for real-time, automatic identification of information in the supply chain of any company, anywhere in the world." In 2004 EPCglobal approved a specification for an RFID air-interface protocol, known as the "Class-1 Generation-2 UHF RFID specification" or just as "RFID Gen 2." Other standards are being developed by ISO/IEC and the European Telecommunications Standards Institute. These protocols address problems in electromagnetic interference, sensitivity to nearby metallic objects, range of data transmission, and the use of multiple readers in close proximity.

Business value proposition

RFID tags are mechanisms for automatically detecting and identifying an object from a distance. The concept bears some similarity to that of bar codes, but RFID tags can be read through an opaque cover without being in direct line of sight, and can have the information upon them changed remotely. RFID tags may be read unobtrusively at a distance, so they could eventually lead to systems allowing shoppers just to wheel their purchases past a

checkout, still in their shopping cart, and get an accurate accounting. They are also an effective theft-prevention device, since tags may be hidden inside products, with readers placed at all exits.

RFID technologies have been used in a variety of ways. Active tags that contain their own power source are frequently used by motorists who wish to pass quickly through toll booths without stopping to pay with cash. The tag's identification number is collected by a reader on the toll booth, and the toll is automatically debited from an established account or charged to a credit card.

The decreasing cost associated with the passive tags has led to their use in a growing number of areas, for example the organizers of marathons frequently have the athletes attach RFID tags to their shoes so that their "real times" can be recorded (in large events such as the Boston and London Marathons some runners do not actually pass over the start line until several minutes after the starter's gun has been fired, due to the number of runners ahead of them in the crowd). Business applications using RFID are also growing, including embedding them in credit cards so that the card can simply be waved in front of a RFID reader. Casinos embed RFID tags inside their chips to prevent the use of counterfeit chips in their establishments, placing readers at the gambling tables.

Large-scale RFID use is dependent upon the dramatic decrease in their cost, and this is dependent upon large orders being placed with their manufacturers. Progress in this area is strong: in 2002 the Gillette Company ordered 500 million RFID tags, and Wal-Mart, the world's largest retailer, has moved toward 100% RFID compliance by vendors. Both of these initiatives indicate the strength of corporate confidence in this technology. Companies are placing RFID tags on pallets (with readers on forklifts) and on packing cases, as well as actually embedded with items.

Civil liberties organizations and political groups have raised concerns about the use of RFID. Central amongst these concerns is the ownership of personal information, and the ability of RFID readers to track and profile individuals. The United States and the European Union are investigating the possibility of embedding RFID tags into their citizens' passports and driving licenses.

Summary of positive issues

RFID technologies allow rapid remote identification of labeled objects. The data in the device can be changed as required. Passive tags do not require an internal power source. Tags are easy to place on objects. Protocols and standards are being developed. RFID tags can withstand a large range of ambient conditions. Unlike with barcode systems, the readers do not have to be in line-of-sight proximity.

Summary of potentially negative issues

RFID tags contain a relatively small amount of data. Costs are relatively high compared with older solutions. It is difficult to ensure that a tag has been read, and especially that all tags have been read when processing collections of labeled objects, and it requires extra receipt acknowledgements to be performed. Standards and protocols are still evolving. There are controversial social issues associated with the use of RFID in the public domain. It is difficult to ensure that tags are deactivated after they have served their legitimate purpose.

References

- H. Stockman (1948). "Communication by means of reflected power," *Proceedings of the I.R.E.*, pp. 1196–1204.
- J. Landt (2005). "The history of RFID," *IEEE Potentials*, Volume 24, Issue 4, pp. 8–11.

- EPCglobal Inc. (2005). "EPC radio-frequency identity protocols Class-1 Generation-2 UHF RFID conformance requirements," http://www.epcglobalinc.org.
- D. Ewalt (2003). "Gillette orders 500 million RFID tags," *Information Week*, January 6.

Associated terminology: Bar code, Business process re-engineering.

Robotics

Foundation concept: Artificial intelligence.
Definition: Robotics is the branch of computer engineering covering the development of computer-operated devices that perform useful physical tasks.

Overview

The idea of a robotic servant that can perform tasks for its human owner has a long and involved history. Humans have been attempting to create robots since the eighteenth century when inventors created "automata" based upon clockwork systems and metal-cam "memory" devices, very much in the way that a musical box or a player-piano worked. However, the term "robot" did not come into use until the beginning of the twentieth century and it is generally acknowledged that the Czech playwright Karel Čapek coined the term, derived from the Czech word for serf or servant.

It was not until the 1960s that working robots started to appear. While these early machines were called robots, they were far from the science-fiction vision of a robot portrayed as a fully operational, thinking, and super-human, if emotionally stunted being (the primary example seems to be Robby the Robot from the 1956 film *Forbidden Planet*). In 1961 General Motors deployed the *Unimate* robot, which was essentially a 4000 lb hydraulic arm attached to a computer. Unimate moved castings to a platform and then performed spot welds upon car bodies.

Since the 1960s computer scientists and artificial intelligence (AI) researchers have worked to improve the flexibility, autonomy, and dexterity of robotic systems. Robotic arms have evolved to be multi-jointed with hands and fingers enjoying seven degrees of freedom, and having tactile sensors on the fingers that can be used to regulate the pressure used to grasp items.

Robotics research at laboratories such as the MIT AI lab has created robotic systems that utilize a variety of technologies to make them more human, exhibiting the human characteristics of adaptability, creativity, and initiative. These are amongst the characteristics required to differentiate "intelligent" robotic systems from robotic systems that can merely perform tasks as defined by their programmers.

At the MIT AI Leg Lab researchers have created robots that can run and perform somersaults, demonstrating agility and balance. They have also created robots that can find their way around the corridors and rooms of the laboratory, demonstrating their visual and spatial reasoning capabilities. Some of the robots in the laboratory can communicate with the people whom they meet in the building and provide assistance to visitors, demonstrating their skills in understanding natural language, speech recognition, and reasoning.

Not all robots are human-like in design, and researchers such as Professor Rodney Brookes at MIT have created smaller robots designed to work in hostile environments, such as those found on the planet Mars. Brookes's robots, such as *Boadicea* and *Genghis*, provided the basis of many of the technologies used in robots such as the NASA Mars Exploration Rovers *Spirit* and *Opportunity*, which touched down on Mars in 2004.

At the outset of the twenty-first century, robots may not be able to make our beds, cook our breakfast, and drive the children to school, but robotic help systems for the home are available from several vendors, including Honda's ASIMO P3, which can navigate around a home and supposedly perform basic household tasks such as fetching the newspaper from the front garden. However, those who expect robots to provide any real independent assistance in the near future are doomed to disappointment.

Much has also been written about a new class of robotic systems termed "nanotechnology," that is technology whose size is measured in nanometers or one-billionths of a meter and hence are built at the molecular level. While the theory holds great promise, the reality of nanotechnology robots being used to perform heart operations or other procedures in the human body remains a considerable way off.

Business value proposition

The vision of genetically based humanoid androids, such as those depicted in the film *Blade Runner*, capable of working in the mines on Mars is at present still only science fiction, and is virtually certain to remain so. The use of robots within corporations is currently limited to industrial robots that work on and around assembly lines. Manufacturers such as FANUC, ABB, Panasonic, Motoman, Kuka, and Nachi provide a wide range of equipment for customers such as automobile companies.

The technology associated with industrial robots is primarily focused upon providing precision performance in processes such as welding. The adoption of a robotic system also requires a company to build a business case that examines operational issues such as the system's ability to reduce work-cycle times, operate in small work areas (so that more robots can operate together on the assembly line), and economic issues such as operational costs incurred by the consumption of electricity and maintenance, and the expected useful lifetime before new technology renders an expensive robotic workforce obsolete.

The military use robotic systems to operate in high-risk environments such as examining potential bombs or hazardous materials. They have spent considerable amounts of money attempting to develop automated fighting machines to replace or assist their human soldiers.

Summary of positive issues

Robotic systems have been researched since the 1950s and a large literature has developed. Industrial robotic systems are widely available for manufacturing. Robots can possess the capabilities of walking, communicating, and vision processing. Robots are in use in hostile and inaccessible environments such as volcanic craters and Mars.

Summary of potentially negative issues

Intelligent robotic systems are typically programmed to work in limited domains, and their level of intelligence is debatable. Industrial robots need to have their processes defined completely for them.

References

- Computer History Museum, 1401 N. Shoreline Blvd, Mountain View, CA 94043, USA.
- http://marsrovers.jpl.nasa.gov/overview/.
- D. Kortenkamp, R. Bonasso, and R. Murphy (1998). *AI-based Mobile Robots: Case Studies of Successful Robot Systems* (Cambridge, MA, MIT Press).

Associated terminology: Neural networks, Machine learning.

RSS (Really simple syndication)

Foundation concepts: Internet, XML.

Overview

Really simple syndication, RSS, was originally introduced in a simple form by Netscape in 1999, and has since undergone a series of modifications and improvements. As a result of this, several different versions are in use today, and the acronym is actually defined three ways, as RDF site summary, Rich site summary, and Really simple syndication. The last is the most used meaning.

Syndication is a process traditionally used in the newspaper industry when content is used by a variety of outlets. For example, a well-known columnist may have their work printed in many different newspapers at the same time, and nearly all popular comic strips are published this way. Web syndication is a mechanism through which an individual may subscribe to content providers, and automatically receive regular updates from them. RSS is used not only for news stories, but also for *blog* sharing, and even pictures and music. RSS readers collect content and place it in a folder for viewing, displaying an icon to notify the user that new items are available. Generally, the whole item is not copied to the folder, but rather a summary together with a hyperlink to the full story.

Business value proposition

RSS provides any entity that deems its information or opinion to be worthy of reading by a subscriber with a semi-standardized mechanism for subscription management and distribution. RSS viewers (client receivers) are universally available, so there is no need to distribute specialist software to subscribers. Content may be made available in one of two ways, either by setting up an individual or corporate RSS server, or by registering with an *Aggregator*. Providing an RSS service is suitable for a large well-branded company whose feeds will be sought out by subscribers, but unsuitable for relatively unknown providers whose sites may simply not be noticed. An aggregation service collects together feeds from a variety of sources in one well-known place, where potential subscribers may browse at leisure and see whatever attracts their interest.

The business model associated with RSS feeds is based on allowing users to sign up for information updates in specific areas of interest and then pushing information out to those users. This builds a strong relationship between the provider and the subscriber since the subscriber has to go via the hyperlink to the story's originating site to read the full article. This is advantageous in that it allows providers to understand and manage their content better, focus the content upon specific audiences, and, of course, focus advertising and other revenue-generating aspects of their sites where applicable.

RSS technology has drawn a lot of attention because it overcomes the limitations of email as a distribution channel, on which similar content is likely to be blocked as spam or just ignored. Users no longer find signing up to email distribution lists an attractive choice due to the widespread abuse of email and the bland nature of the text delivery. RSS is also attractive to businesses since it leverages information that they have already provided. The system may be automated, and requires relatively low resource overheads to manage.

Summary of positive issues

RSS is an easy-to-use mechanism for syndicating information. RSS readers are widely available. Some browsers have RSS readers built into them, and there are many vendors supplying commercial tools and solutions. Open-source and free RSS readers and tools are also available. RSS leverages existing information for low cost, and provides organizations with the ability to have a better understanding of their customers and

leverage that relationship. RSS technology is relatively straightforward.

Summary of potentially negative issues

RSS technology is still evolving and several technology groups are attempting to provide direction for future standardization.

Reference

- B. Hammersley (2003). *Content Syndication with RSS* (Sebastopol, CA, O'Reilly Press).

Associated terminology: Web services, Client–server.

Sarbanes–Oxley Act of 2002 (SOX)

Definition: The Sarbanes–Oxley Act of 2002 was enacted to "protect investors by improving the accuracy and reliability of corporate disclosures made pursuant to the securities laws, and for other purposes" (Sarbanes–Oxley Act of 2002, Report 107–610).

Overview

The Sarbanes–Oxley Act of 2002 was a response to the financial reporting and disclosure problems associated with companies such as Enron, whose 2001 collapse was the largest bankruptcy in US history. This large and complex act pertains to corporate governance practices in public companies and contains eleven titles:

1. Public company accounting oversight board
2. Auditor independence
3. Corporate responsibility
4. Enhanced financial disclosures
5. Analyst conflicts of interest
6. Commission resources and authority
7. Studies and reports
8. Corporate and criminal fraud accountability
9. White-collar crime penalty enhancements
10. Corporate tax returns
11. Corporate fraud and accountability

While the act is wide-reaching in scope and focuses on corporate and executive accountability for financial data, the maintenance of internal control structures, and the role of accounting firms in the audit process, for the CIO or IT professional it does not contain any specific systems requirements and in fact never even mentions the word computer in its 66 pages. However, it is clear that technology and information systems will be central to corporate compliance with the act.

The eleven titles of the act contain 69 sections, several of which are regarded as key from a CIO's perspective, including the following.

Title II – Auditor Independence, Section 201: *Services Outside the Scope of Practice of Auditors* "(g) Prohibited Activities . . . (2) financial information systems design and implementation." Termed "non audit services," accounting firms are banned from providing systems consulting, and as a consequence most accounting firms have spun off their consulting practices, which now have to maintain independence from the process of implementing controls and performing audits.

Title III – Corporate Responsibility, Section 302: *Corporate Responsibility for Financial Reports* "(a) . . . that the principal executive officer or officers, or persons performing similar functions, certify in each annual or quarterly report filed or submitted under either such section of such Act that (1) the signing officer has reviewed the report; (2) based upon the officer's knowledge, the report does not contain any untrue statement of a material fact or omit to state a material fact necessary in order to make the statements made, in light of the circumstances under which such statements were made, not misleading . . . (4) the signing officers (A) are responsible for establishing and maintaining internal controls; (B) have designed such internal controls to ensure that material information relating to the issuer and its consolidated subsidiaries is made known to such officers by others within those entities, particularly during the period in which the periodic reports are being prepared; (C) have evaluated the effectiveness of the issuer's internal controls as of a date within 90 days prior to the report." The effect of this aspect of the act is to enforce strict controls upon access to data and reporting systems. The mechanisms through which this can be done are varied, and the use of modern ERP-type systems or packages as opposed to legacy systems typically eases

the compliance officer's task because many of the security barriers have been constructed into the system by the vendor.

Title IV – Enhanced Financial Disclosures, Section 404. *Management Assessment of Internal Controls*, "(1) State the responsibility of management for establishing and maintaining an adequate internal control structure and procedures for financial reporting," together with Section 409, "(1) Real Time Issuer Disclosures – Each issuer reporting under section 13(a) or 15(d) shall disclose to the public on a rapid and current basis such additional information concerning material changes in the financial condition or operations of the issuer, in plain English, which may include trend and qualitative information and graphic presentations, as the Commission determines, by rule, is necessary or useful for the protection of investors and in the public interest." The development of policies and procedures to maintain internal controls typically falls under secure identity management (SIM), whereby all security points for data and personnel are assessed and the required level of security is implemented. The primary vehicle through which Section 409 (1) is approached is the posting of information upon a web site accessible via the internet. Additionally, document-management systems are also valuable resources to provide facilities such as versioning, archiving, and managing heterogeneous file types (audio, video, email, text, web documents, etc.).

Title VIII – Corporate and Criminal Fraud Accountability, Section 802. *Criminal Penalties for Altering Documents*, "(a) (1) Any accountant who conducts an audit of an issuer of securities . . . shall maintain all audit or review work papers for a period of 5 years from the end of the fiscal period in which the audit or review was concluded. (2) The SEC shall promulgate, within 180 days after adequate notice and an opportunity for comment, such rules and regulations, as are reasonably necessary to the retention of relevant records such as work papers, documents that form the basis of an audit or review, memoranda, correspondence, communications, other documents, and records (including electronic records) which are created, sent, or received in connection with an audit or review and contain conclusions, opinions, analyses or financial data relating to such an audit or review which is conducted by any accountant who conducts an audit . . ."

The retention of data and information by the public company and the auditors is a major aspect of *Information-lifecycle management* (ILM), in which the frequency of use of the data is related to the type and cost of the storage medium utilized for storing that data. Included in this data set is the email correspondence of the organization, the archiving of which needs special concern since the volume of emails may be considerable, thus requiring specialist storage services. Additionally, special techniques will be needed to capture other forms of electronic communication such as instant messaging and text messaging.

In order to comply with these and the other sections of the act, CIOs have utilized a series of frameworks. Primary amongst these are COSO, CobiT, Trust Services, and the ISO 17799 security standard.

The Committee of Sponsoring Organizations of the Treadway Commission (COSO) is an organization founded in 1985 to examine the "causal factors that can lead to fraudulent financial reporting" and to "develop recommendations for public companies and their auditors." The COSO framework was developed in the 1990s to manage the SEC's demands for internal audit controls and has been adopted by SOX-compliance managers to meet the act's demands. The framework is composed of five dimensions,

1. Control environment
2. Risk assessment
3. Control activities

4. Information and communication
5. Monitoring

While these dimensions are very useful, they are not specific step-by-step instructions that can be implemented. This is by design, since they were intended to be guidelines for a wide variety of companies and situations; there can never be a specific set of "one-size-fits-all" instructions on how to be compliant. The COSO framework is typically used with other frameworks such as those provided by CobiT, or by trust services that are more specific in nature and help organizations focus upon technology-related issues.

The Control Objectives for Information and Related Technology (CobiT) framework was developed by The Information Systems Audit and Control Foundation to assist organizations to manage compliance through the provision of good practices that work toward balancing risk, control, and technical issues.

The control of access to financial systems and financial data is at the very center of the SOX act and the control responsibilities are typically divided into three areas: *General* control issues such as data management, disaster recovery, and the physical and logical security of resources; *Company*-level controls, including such issues as implementing codes of conduct, policies to prevent fraud, and operational governance policies (these controls establish the "control environment" within the organization); and *Application* controls which are controls actually embedded in the organization's processes in order to ensure their correct use (these include such controls as authorization approvals and approval routings). These three aspects of control are then considered and developed further in terms of the risk criteria with each process and task, the monitoring of the controls, and the communications that surround the management of control functions.

Trust services are a set of "professional assurance and advisory services based upon a common framework to assess and address the risks and opportunities of IT" developed through the American Institute of Certified Public Accountants. The foundation for trust services encompasses four sets of principles and criteria: policies, communications, procedures, and monitoring, together with five sets of principles and criteria developed for policy implementers. These consist of attributes and controls in the areas of security, availability, processing integrity, online privacy, and confidentiality.

Business value proposition

The business value proposition of the SOX act is to provide evidence for the shareholders, employees, suppliers, vendors, and others associated with an organization that the financial control structures are in accordance with strict audited accounting standards, and that a financial controls structure is indeed in place.

Summary of potentially positive issues

The SOX act helps investors in public companies to have confidence that the financial aspects of the company will adhere to the stringent requirements of the act, and that, should an organization not adhere to the act, then the executives and board of directors will be held accountable. The act makes organizations consider their strengths and weaknesses in terms of their control structures and financial-reporting systems.

Summary of potentially negative issues

SOX is frequently discussed in terms of the issues surrounding Y2K compliance, except that failure to meet Y2K compliance did not potentially result in the CEO, CFO, and others in the organization going to jail. SOX has been criticized for being extremely expensive and hard to implement, and for requiring large amounts of

corporate resources and commitment. This has been especially hard on many smaller public companies and has resulted in companies delisting from the public markets and returning to being privately held entities. It has also been suggested that, as a cost of business, it has acted as a barrier to entry for companies wishing to locate in the United States or use the US public stock markets to raise capital.

References

- The IT Governance Institute (2004). *IT Control Objectives for Sarbanes–Oxley* (Rolling Meadows, IL, The IT Governance Institute), www.itgi.org.
- The IT Governance Institute (2000). *CobiT Framework*, 3rd edn. (Rolling Meadows, IL, The IT Governance Institute), www.itgi.org.
- The IT Governance Institute (2000). *CobiT Executive Summary*, 3rd edn. (Rolling Meadows, IL, The IT Governance Institute), www.itigi.org.
- Committee of Sponsoring Organizations of the Treadway Commission (2004). *Integrated Control Framework*, Volumes I & II, www.iacpa.org.

Associated terminology: Internet, ILM, Email, Instant messaging, ISO/IEC 17799.

Scalability

Foundation concept: Software development lifecycle.
Definition: The ability to expand a system without making significant changes. Getting more of the same results by using more of the same solution.

Overview

Scalability is a very simple concept. If a merchant needs to store 12 000 cubic feet of non-perishable merchandise, they might decide to build a 3000 square foot one-storey warehouse. With merchandise stacked 8 feet high, covering 50% of the floor-space (to allow access), the size matches perfectly. If that merchant expands, and needs to store 24 000 cubic feet of merchandise, simply building another 3000 square foot warehouse will solve the problem equally well. If they eventually need to store 120 000 cubic feet, the process of building a new 3000 square foot warehouse, applied ten times, will yield a working solution. The idea of building warehouses to store merchandise is a *Scalable solution*.

If a publisher decides to produce a small encyclopedia with 4000 entries, they may decide to print it as a single large book, with 1000 pages printed on very lightweight paper, in the style of a large dictionary. If the encyclopedia proves popular, and a second, extended edition with 5000 entries is planned, they could simply expand the book to 1250 pages and use a slightly stronger spine. However, the same expansion process can not be applied too many times. It is almost impossible to produce a 10 000-page book: the binding falls apart, and it is too heavy to lift. This is not a scalable process. Eventually a jump to new technology is required.

For book production, the jump to new technology is simple: just publish the book in more than one volume. The analog for computer systems, both hardware and software, is usually not so simple.

A single desktop PC, with a suitably robust operating system, when acting as an enterprise web server can support a surprisingly heavy load. A thousand gigabytes of data and a million hits per day are realistic expectations. However, there are limits. The standard IDE/EIDE disk system supports a total of four devices, and even wide SCSI is limited to 15, so disk storage can not be expanded without limit. A T1 internet connection provides 193 000 bytes per second of bandwidth: as usage increases, that provides an absolute upper limit to capacity. Simply using two computers instead of one does not provide a viable solution. How are accesses to be distributed between the two computers? How can one ensure that the

two computers have consistent versions of the data at all times?

In a similar vein, the simple techniques of programming used by beginning programmers are also very often not scalable. The commonly understood methods for searching data sets for example are linear with time, which means that, as the amount of data to be searched grows, the time required to search it grows in direct proportion. This sounds like an obvious observation, but is in reality a critical problem that can be fatal to ill-conceived software projects. Under light load, testing with 1 MB of data might reveal that the average search takes one tenth of a second, and that sounds quite acceptable. But when the system goes live with perhaps 10 GB of data (10 000 times as much), that one tenth of a second grows to one thousand seconds, or a quarter of an hour, reducing the system to a maximum of 96 agonizingly slow accesses per day. (See *Complexity*, *Algorithm*, and *Data structure* for more depth.)

Ensuring scalability requires intimate knowledge of algorithm analysis, data-structure design, and network protocols, and requires familiarity with the vast corpus of human experience that 60 years of intense study has created. Of course, an intelligent programmer could in principle rediscover all of this knowledge for himself or herself, just as an intelligent blacksmith could in principle independently reinvent the Apollo lunar-landing module.

Business value proposition

Lack of scalability in software solutions is the hallmark of the untrained programmer. It is often possible to learn a little programming by reading a book or two, and then produce small-scale solutions, perhaps using Visual Basic or JavaScript, that stand up to simple test scenarios. When software has to support literally millions of accesses every day, with perhaps hundreds happening concurrently, and remain operational 24 hours a day, 7 days a week, more sophisticated design is needed. It takes many years for a programmer to gain sufficient expertise in such areas.

Summary of positive issues

There exists an extensive body of knowledge pertaining to the problems of scaling software and hardware systems. Commercial systems that will scale to significant levels are available for common problems and tasks.

Summary of potentially negative issues

Failure to understand the scale to which a task may grow can result in solutions that fall short, cause system failure, and be detrimental to the business. It is expensive to train IT personnel to the levels required in order for them to plan, design, build, and maintain systems that scale significantly.

Reference

- K. Hwang (1992). *Advanced Computer Architecture: Parallelism, Scalability, Programmability* (New York, McGraw-Hill).

Associated terminology: Reliability, Computability.

Secure

Foundation concept: Encryption.

Definition: Secure web pages and secure transactions are transmitted over the network with an encryption method that ensures complete confidentiality even if the entire interaction, including key exchange, is intercepted.

Overview

When reviewing personal records, or making an online purchase with a credit card, or when privacy is a concern for any other reason, the internet can be an alarming communications medium. There are so

many opportunities for communications to be intercepted anywhere along what may be a very tortuous path between browser and server. Physical security, preventing transmissions from being intercepted, is impossible, so *Encryption* must be used to ensure that intercepted communications will be unreadable.

The *Secure-sockets layer* (SSL) and other similar technologies provide a universally standard means of encryption that gives complete protection regardless of how badly compromised the communications network may be. Traditional encryption methods always had one weak spot: the sender and receiver must agree on a key to be used for encryption and decryption. If the sender and receiver are never physically in the same location, the key must be transmitted, and could easily be intercepted, resulting in all communications being readable. If the same key is used more than once, it provides eavesdroppers with a much greater chance of decoding messages by analytic means.

Newer developments, owing much to the *Diffie–Hellman* technique, completely remove this problem, with a cleverly designed protocol that allows sender and receiver to agree upon a secret key that can not be determined even by an eavesdropper who hears the entire conversation. SSL and other systems use these secure key-generation methods to securely create a totally new encryption key for each communications session, so there is also no risk associated with the reuse of old keys.

A *Secure web page* is one that is transmitted from the web server to the user's browser using this technology, so that its content is guaranteed to be readable only by the intended recipient. A *Secure transaction* is an entire online "conversation" between a web browser and the server, every stage of which is transmitted securely. *SET* (Secure Electronic Transaction) is a particular protocol for secure credit-card transactions that was introduced jointly by Visa and Mastercard, which works by the same basic methods.

Secure communications do offer some protection against identity theft, but are no protection if identity theft has already occurred. If a criminal can successfully pretend to be another person, having stolen their password, they will be able to engage in secure transactions under that identity.

Business value proposition

Security is one of the fundamental aspects of computer and network design, and the technologies that support the desired security level need to be designed into the system at the outset whenever possible. The design of the security component is the responsibility not only of the network administrator and the chief security officer but also of the CIO, the CEO, and the board of directors. For example, should a security breach occur or if customers don't see the user interface that they understand and are familiar with (e.g., the padlock that signifies the secure-sockets-layer-based connection), it is less likely that customers will use the site. However, the level of security needs to be based upon the circumstances and basis of the business and transactions occurring: a major bank will be clearly required to have a higher level of secure technology than a small trader, and a purely informational site with no financial transactions may need none.

Summary of positive issues

Established and well-understood technologies exist to ensure the security of the data transmitted on the internet between companies or individuals. Technologies exist to provide security at a variety of levels.

Summary of potentially negative issues

Even though technologies exist, security can be maintained only if the technologies

are implemented and used correctly. No security system can be expected to be perfect, but processes and measures must be in place to identify immediately any breakdown in security that has occurred, as well as reporting any unsuccessful security breaches so that weak spots may be identified before it is too late. Security should not only be reactive to incidents, but should also be a continuous background presence. It is important to assess carefully the true value of security policies in action, since some seemingly valid policies can have a negative effect; for example, the common requirement to change passwords every month results in users having meaningless unmemorable passwords that they need to write down, and a written password is much easier to find.

Reference

- K. Day (2003). *Inside the Security Mind: Making the Tough Decisions* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Security, Internet, Protocol.

Security

Foundation concept: Secure.

Definition: An information system's security pertains to the task of understanding those aspects of an organization's information system and processes that may be vulnerable to unauthorized intrusion, abuse, or attack.

Overview

Security of information systems is primarily concerned with access, physical as well as through the network. Good security commences with an appraisal of access rights and then enacting physical and technological solutions to guarantee that security is achieved.

Physical security is frequently overlooked, but all access to secure and non-secure areas needs to be considered. For example, if a trader at an investment bank left their computer logged in to the trading system, a cleaner with legitimate access to the area at night could gain access to that system, send rogue emails, place trades, or acquire proprietary information about an IPO pricing strategy. Similarly, all technical correspondence should be shredded or disposed of in a secure manner, because even seemingly irrelevant information can be useful to someone with bad intent, e.g., the CEO's internal email address or the CFO's system ID could be a starting point for a hacker's attack. The physical security of the computers in the organization needs to be examined since it is not unknown for people to load deliberate software bugs that copy log-in data or corporate information and then transmit it secretly to the attacker (or an accomplice) via the internet.

Technology clearly can play a large part in securing an organization's systems. For example, the use of encryption, thin clients, and password-protection systems, requiring the operating system to time-out and disconnect users after short periods of inactivity, having servers disable workstations and clients on the network after normal work hours, and using biometrics to log in to systems can all play a part in securing the systems from intrusion.

Security concerns also emanate from external entities that attempt to intrude upon the network and its contents. Attacks come in many forms, including attempts to hack into the network, or crack the security codes associated with a network. Alternatively, networks may come under attack from *Viruses*, *Worms*, *Trojan horses*, and *Denial-of-service* attacks. Defenses against each of these can be mounted, including the creation of software and hardware barriers such as *Anti-virus* software, *Firewalls*, *Proxy servers*, and the use of networks that are not capable of connecting to the

internet unless *Network address translation* is used. Other plagues that are usually not fatal but are detrimental to corporate systems' performance include *Spam*, *Pop-up ads*, and *Phishing*. These can be addressed through the use of spam filters and of software that removes spyware and prevents pop-ups from being executed. Phishing requires corporations to be proactive and to help customers affected by such attacks, and to have *e-Commerce* policies that minimize their effectiveness.

While computers and networks can seem secure, frequently security measures are undermined by individual weaknesses, such as leaving a laptop computer with no password protection in a car that is stolen, downloading data to unauthorized portable computers that are taken out of secure premises, using wireless systems that beam the signal through office or hotel walls, and using laptop computers on airplanes where corporate information or client information can be surreptitiously viewed. Even cell phones and cell phone conversations need to be considered as part of the security umbrella of an organization and solutions found for these areas of concern.

Corporate IT security concerns are addressed by many laws, including the CAN-SPAM Act of 2003, which enacts requirements on commercially motivated email; HIPAA (the Health Insurance Portability and Accountability Act, 1996) that legislates the security requirements associated with digital medical systems; the UK Computer Misuse Act of 1990, which forbids unauthorized access to computers and the data held on them; and the US Identity Theft and Assumption Deterrence Act of 1998, which makes most phishing criminal. These and other laws come with strong penalties to deter potential offenders, but the nature of the internet makes many offenders difficult to track down and many companies do not wish to prosecute every small attack because this would potentially have the negative effect of reducing customer confidence in their system and defenses as well as making the company a potentially attractive and challenging target for other attacks. Laws are also not "practically" helpful in the event that an attack successfully crashes the corporate network or when private data has been stolen; the damage has already been done.

Business value proposition

Security is an aspect of a corporate IT organization that is usually regarded as a sunk operational cost; while the prevention of intrusion or attack is valuable, it does not actually help raise the top line of the business (revenue) unless a company uses it as a selling point. In the last century banks used to have large safes visible to customers entering the bank and this encouraged their customers to think that their deposits would be safe. In the modern technological organization it is becoming more and more a corporate imperative to ensure the security of their systems and, more importantly, the security and privacy of their customers' data.

Many companies have created a position of Chief Security Officer (CSO) responsible for all security concerns, including physical and intangible assets. This is an important position, especially in a public company that has to adhere to corporate governance legislation such as Sarbanes–Oxley. CEOs, CIOs, and CSOs all need to be aware of technology-related security issues and these may influence decisions regarding corporate strategy. For example, it is vital to understand the legal consequences of outsourcing information-based services to countries with weak IT laws and protection. One approach that has been used successfully has been to adhere to ISO 17799:2000, which is a code of practice for information-security management and offers guidelines on the development of policies and good practice in this area.

The need for security within an organization's IT systems is a critical factor in maintaining stakeholder (customer, vendor, shareholder, and governmental) confidence in the organization. However, many organizations are unable or can not afford to staff a full-time CSO position and have instead used the services of vendors who specialize in security issues to perform security assessments.

Summary of positive issues

Many of the security concerns for organizations are well known and procedures and policies are available to address them. The laws of many countries cover the majority of abuses relating to systems security and specify significant penalties for offenders. Organizations can appoint a CSO to head the security initiative. There is an ISO code of practice for information security management to guide organizational policies. There are many vendors and consultants worldwide that provide security services, tools and resources.

Summary of potentially negative issues

Security problems continue to surface in all areas associated with corporate (and personal) computing environments: hardware, software, the physical infrastructure, and human resources are all vulnerable. Thus the target for CSOs continues to move, requiring continuous investment in resources to combat these problems. The legal framework acts as a deterrent but can not prevent criminal activity. The ISO code of practice for information security management is a set of guidelines, not a manual of solutions.

Reference

- K. Day (2003). *Inside the Security Mind: Making the Tough Decisions* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Encryption, Internet, Protocol.

Server

Foundation concept: Network.
Definition: Servers provide the resources to a network; these resources are accessed by end users through client devices.

Overview

A *Server* may be any device that provides computational, data-processing, storage, or other services to other *Client* systems. The usual model is that a server has a long-term passive network presence, providing services to client systems only on request, whereas a client is an independent component, actively requesting services when it (or its user) sees fit.

A server can be as simple as a personal computer acting as a file system or webpage host for a small group of clients, or as complicated as a cluster of processors hosting complex software applications such as an ERP or distributed database for thousands of clients over a global network. Servers provide centralized computing services that may be secure, scalable, and regulated. The centralization allows network administrators to better protect their organization's systems from unauthorized access, attack, and failure than if an organization had a wide variety of resources spread over the business units. The scalability of the system comes from the modular design of the typical server system, to which computing power, in terms of the number of processors, can be added as the demand for system resources grows.

Operating systems suitable for supporting large-scale servers, such as Unix (and Linux, a variety thereof) and Microsoft server products, allow the user demand to be balanced across multiple servers if demand increases, and thus help to optimize performance levels. Specialized servers are also generally incorporated into the overall design of a corporate server-based architecture; these, for example, act to provide data files (file servers), web pages

(internet servers), and applications (application servers), and to relieve the load on the main server which is interacting with its clients and determining how best to service requests (the main server in a Microsoft-based system is termed the primary domain controller).

Business value proposition

The server concept allows corporations to develop flexible, scalable computer systems that run across networks. The associated client–server technologies allow greater flexibility in terms of configuration, the variety of applications that can run on the platform, and control of user access compared with traditional mainframe *dumb-terminal* configurations. Server technology may be configured to support a wide array of applications, including specialist servers such as web servers, that address the computational needs of particular groups of users.

Server technology allows companies to create and administer a strong security policy based on the centralization of data and applications in a small number of server locations. This allows the CIO, the network administrator, and the Chief Security Officer to focus resources on the problem of system security. Security policies may be implemented more efficiently on a server-based architecture than when corporate IT operations are distributed amongst the business units with no central control.

Server architecture also allows the IT organization to standardize and regulate its software systems rather than allowing individuals and business units to make their own software choices. Server technology is mature enough that good cost-of-ownership decisions can be made centrally by the IT organization. A key aspect of this calculation is centered upon the software license fees, which may be based on the number of users and the processing power of the system. It is therefore important for the CIO to ensure license productivity and that decisions affecting the return on investment are correctly considered.

Summary of positive issues

Servers facilitate strong centralized control of an organization's computing resources. Servers provide a flexible, scalable, secure architecture through which to deploy computing resources. There is a choice of operating systems upon which servers can run. Servers can run major applications such as ERP systems or act as file servers for a company's web-page service.

Summary of potentially negative issues

Large server architectures require substantial expertise to run effectively. Dependence upon a central system can leave the users vulnerable should that system fail.

References

- J. Chellis, C. Perkins, and M. Strebe (2000). *MCSE: Networking Essentials, Study Guide* (Alameda, CA, Sybex Press).
- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).
- D. Groth (2003). *A+ Complete* (Hoboken, NJ, Sybex–John Wiley and Sons).

Associated terminology: Client, Client–server, Web services, Peer to peer.

SIIA (the Software and Information

Definition: The Software and Information Industry Association is a trade association representing members of the software and digital-content industry.

Overview

The Software and Information Industry Association (SIIA) is a trade association for the software and digital-content industry. The association provides a variety of services, including representing its members' interests at the governmental level,

protecting the intellectual property of members, promoting business development, and providing educational services at a corporate level and as a public service.

The association is divided into four divisions: Software, Content, Education, and Financial Information Services, through which a common policy formulation is developed and put forth in the form of white papers, briefings, and other media. Each division develops the agendas that are important to its members, and, through working groups and studies, develops policy in each area. For example, the Software division has working groups that consider initiatives such as web services, open-source systems, and software as a service.

The SIIA is also active in the development and enforcement of anti-piracy policy, and has created an anti-piracy department to focus upon this area (the division was formerly known as the Software Publishers Association). The SIIA also offers a course that leads to the Certified Software Manager (CSM) qualification, which is aimed at network managers and technology professionals.

Business value proposition

The SIIA provides a forum through which common subject matter can be discussed and policy positions ultimately developed. These policy positions can then be put before legislators by the SIIA to help provide advice and education. The body also provides industry intelligence to members through briefings, reports, and conferences.

Summary of positive issues

The SIIA has a large membership with a wide range of interests. Membership fees are on a sliding scale depending upon the revenues of the company joining. The SIIA is influential in providing representation at the governmental level and provides members with educational resources. The SIIA is proactive on anti-piracy issues and provides certification for IT professionals pertaining to piracy issues.

Summary of potentially negative issues

The requirement to join an umbrella organization such as the SIIA is optional and it can not be seen as the only voice in the technology industry.

Reference

- SIIA Main Office, 1090 Vermont Ave NW, Sixth Floor, Washington, D.C. 20005, USA; http://www.siia.net/.

Associated terminology: ACM, BCS, IEEE, W3C.

Software development lifecycle

Foundation concept: Software.

Definition: The software development lifecycle is the series of processes through which a concept is transformed into a program.

Overview

The *Software development lifecycle* (SDLC) covers the transformation of a concept into a verified and validated software product. *Verification* is the process of ensuring that the system created matches its specification; *Validation* is the process of ensuring that the specification satisfies the customer's requirements.

An SDLC can be designed to consist of any combination of development methods as long as the resulting code matches the specification to a degree of rigor that satisfies the customer. The weakest form of lifecycle is one in which somebody verbally proposes an idea for a program and then just starts to code directly on the computer, running the resultant code and testing it until they are satisfied. This method is clearly not scalable and makes proper verification impossible since there is no specification to measure the system against.

At the other end of the development spectrum are *Formal methods*, which can be considered a development lifecycle based upon mathematical methods, first for specifying the program to be developed, and then again in applying mathematical-proof rules to refine the program from the abstract specification to the concrete code level.

Between these two extremes lies a wide range of lifecycle models, including rapid prototyping (also known as the *Evolutionary development* model) whereby a prototype is constructed to represent the functionality of the program that is ultimately to be developed. This prototype undergoes cyclical refinement until it matches the user's requirements, at which point the system should then be re-written using more efficient and structured code. (This step, however, is frequently not undertaken.)

Many models based upon the "Waterfall" model have been developed. This approach is based on the unlimited repetition of five basic stages. The stages are requirements definition, system and software design, implementation and unit testing, integration and system testing, and operation and maintenance. In 1988 Barry Boehm advanced a *Spiral model*, which has four repeated phases: determine objectives, alternatives and constraints, evaluate alternatives and identify and resolve risks, develop and verify the next-level product, and plan next phase. The model can be visualized as a "nautilus shell," such that the developer starts in the middle with a review, determines the risk and builds a prototype, considers the prototype in operational issues, and then plans the next stage. This set of process operations is repeated, gradually moving development toward a production-quality system with each iteration of the model.

To assist developers in understanding what level of formality is associated with the methodology they are using, the Software Engineering Institute at Carnegie Mellon University proposed in 1995 the Capability Maturity Model for Software (CMM), which in 2000 became the CMMI (Capability Maturity Model Integration). The CMMI defines parameters and practice levels in a set of models (software, systems, integrated products, process development, and supplier sourcing) that combine to determine which practice level an organization has achieved.

Business value proposition

In order for organizations and individuals to develop efficient, effective programs, each and every time, it is necessary for them to adopt the appropriate SDLC methodology for each project undertaken. The use of a lifecycle model enables organizations to establish internal design standards for their programming. This helps them to be more efficient at measuring their productivity, to be consistent in their training and recruitment policies, and to develop metrics programs in accordance with their processes.

There is a wide variety of SDLC methodologies available to developers, ranging from the formal to the informal. In between the informal and the formal are *Structured methods* such as *Jackson structured design*, which have been popularized and modified by consultants and by industry practices (such as *JAD* and *RAD*). Some SDLC models are the result of the requirements of a particular industry or regulatory body, and contain design criteria necessary for a specific type of software (such as MIL-STD-498, a merger of DOD-STD-2167A and DOD-STD-7935A for military software applications).

The software industry has a vast array of tools, vendors, and consulting firms that support and offer advice on methodologies or parts of methodologies and the management techniques that surround them.

Summary of positive issues

There is a wide array of well-documented and -researched methodologies that range from the formal to the informal. Consulting, tools, and support are available for many methodologies.

Summary of potentially negative issues

Choosing the wrong method can be expensive, time-consuming, and potentially catastrophic, depending upon the project. The professional skill levels required for each vary considerably; formal methods, for example, while being the most rigorous also require extensive rigorous formal training on the part of the software engineer and of any user who has to be able to read the mathematical specification.

References

- www.sei.cmu.edu.
- I. Sommerville (2004). *Software Engineering* (New York, Addison-Wesley).
- R. Plant and R. Gamble (2003). "Methodologies for the development of knowledge-based systems 1982–2002," *Knowledge Engineering Review*, Volume 18, No. 1.

Associated terminology: Structured design, Formal methods, JAD, RAD, Waterfall method.

Software metrics

Foundation concept: Software development lifecycle.
Definition: A well-defined measurement of some aspect of software or software production processes.

Overview

Software metrics have been used in relation to software development and software engineering since the 1960s with varying degrees of success and relevance. The early metrics research of Stroud attempted to measure the skill levels of programmers, whereas Halstead considered the complexity associated with a software program and, by counting the unique number of operators μ_1 and number of unique operands μ_2, together with the total number of occurrences of operators N_1 and the total number of occurrences of operands N_2 in a program, constructed a formula intended to measure the intellectual effort required to produce the program, measured in units of "mental discriminations," of which a competent person is capable of 5–20 per second, according to psychologist John Stroud:

$$\text{Effort} = \mu_1 N_2 (N_1 + N_2) \log_2(\mu_1 + \mu_2)/(2\mu_2)$$

Another metric that has frequently been used is the basic count of the number of lines of code in a program and is an example of a direct metric that does not depend upon any other measurement. This metric has been incorporated into many other models, including Boehm's Constructive Cost Model, known as COCOMO, a model that can be used to estimate the potential costs associated with a software project through the equation

$$\text{Effort} = a(\text{KDSI})^b$$

where effort is measured in person months, a and b are constants determined by the mode of development, and KDSI stands for thousands of deliverable source instructions. There are three modes of development: organic (the team has worked together before, the domain and system are familiar), semi-detached (the team has mixed skill levels, and some familiarity with the domain and the system), and embedded (complex heterogeneous systems development). The COCOMO metric is an example of an indirect metric that involves one or more other metrics.

Both of these metrics have enjoyed wide popularity, regardless of the fact that they can produce wildly different results when applied to the same sample programs. They are intended to work with increasing degrees of accuracy as the developer

moves from the highest levels of design to more concrete implementation designs. However, they also share the disadvantage of being able to produce final effort estimations only once the software has actually been created and the actual total number of lines of code in the system is known. Neither can directly predict the effort required for a new project with any accuracy, which would be a much more valuable metric. It is also problematic that the code developed and the circumstances in which it is developed are never the same twice, making effort forecasts difficult to produce.

There are metrics that attempt to predict the size, complexity, or effort required to create a program on the basis of its requirements, so that they can be used to estimate the costs for a new project. These include Albrecht's *Function Point Analysis*, DeMarco's *Bang Metric*, and Putnam's *SLIM* model, but no single model has been shown to be especially accurate in predicting software-development costs or associated effort, so models need to be used with a degree of caution.

Business value proposition

Metrics have been used in many ways within computing; these include the benchmarking of systems' performance, developing reliability models of systems' performance, quality-of-service metrics, and measurement of algorithm complexity. Owing to the weak statistical base and mathematical underpinning of many metrics, care needs to be taken in the selection of any metric utilized. There is a wide variety of opinion on the validity and usefulness of the various software metrics: Halstead's system has been described as "convoluted" and "unreliable" ("Software metrics: good, bad, and missing," C. Jones, in *Computer*, September 1994), and as "among the strongest measures of maintainability" ("HP-MAS: a tool for software maintainability," P. Oman, University of Idaho Test Laboratory, 1991).

Summary of positive issues

There is a large and established metrics literature. There have been many tools developed to automate the metrics data-collection process. Algorithm analysis is a form of metric that is based upon sound mathematical principles and helps developers understand the complexity of the code they are developing in a formal manner.

Summary of potentially negative issues

There is considerable debate over the worth of many software metrics such as Halstead's, and many metrics are no longer used in commercial systems development because they do not map onto object-oriented design and development methods. Metrics need to be adjusted as the systems they monitor change (e.g., measurements of attributes of Cobol programs can not be equated with the same measurements applied to C++ programs); failure to adapt the metrics will result in wildly incorrect results.

Reference

- N. Fenton (1991). *Software Metrics* (London, Chapman and Hall).

Associated terminology: Benchmark, Algorithm.

Software package

Definition: A software package is an application or a suite of applications and data purchased from a vendor.

Overview

The term *Software package* is typically associated with an application written by a vendor independent from the company purchasing that software. This can range from a custom macro for a spreadsheet application to a full ERP system.

Commercial software packages are generally written by vendors for sale to a wide range of customers (for example, a

word-processing application such as Microsoft Word is a software package for a mass market). While many packages are available to users only in an executable form (binary code), some vendors provide open access to the source code of the software. Such packages are known as "open-source" systems, allowing users to modify the code to meet their own needs. Vendors who provide open-source code access place restrictions on what may legally be done with the code, controlling, for example, whether that code may itself be sold or whether access to it may be provided only for free to other users of the package.

Commercial software vendors who provide only the executable code usually allow programmers to access their code through what are known as *Application program interfaces* (APIs). These are vendor-defined conventions, routines, protocols, and tools for accessing a particular aspect of the system, but not the whole system. The APIs are essential for developers who need to move the package between operating systems or interface it with another application, for example. A third category of software package is known as "freeware" and this is associated with software that is open to use by anyone, potentially with no restrictions on use but usually with the provision that it is not repackaged and resold. A final category or term associated with software packages is "shelfware," which refers to software that is purchased but never used and resides on the shelf of the user.

Business value proposition

Software packages free individuals and organizations from developing their own code or "reinventing the wheel" as it is commonly termed. Clearly it is in very few people's interest to write their own word-processing system or spreadsheet application. At the other end of the spectrum, ERP vendors provide very large systems with a high degree of functionality, and save companies from having to write systems for many activities, including human-resource management (HRM), customer-relationship management (CRM), manufacturing, and supply-chain management (SCM).

While ERP systems are powerful and provide many features, and even though vendors attempt to write industry-specific versions (healthcare, retail, automotive, etc.), a package can never be all things to all people. Independent software vendors fill the void, producing software modules for specialty functions. One such example is the need for wine producers to have a software package to manage the production of wine; the problem of blending and mixing wines to achieve a consistent blend for the vineyard is a niche task, outside of the mainstream products of the ERP vendors. To solve this problem, they would turn to an independent developer, or typically to a company that is certified by the ERP vendor, for a solution. Sometimes, these third-party vendors have "canned" solutions that may be quickly modified to meet a customer's need in a particular area, or else they may have to write a completely new application.

In making a determination of whether to build or buy a system, the question CIOs, business owners, and IT managers need to address is whether or not the system they are considering is core to their business. If it is, then they need to determine whether they should buy a software package, buy and modify a package, or build the system themselves. They also need to consider the operation and maintenance of the system, and whether this will need to be outsourced. If the system is not core, the software should reflect best practices, be easily modifiable, and adhere to industry standards; typically such a solution can be found in a pre-existing package.

Other key issues are the following: will the package run on the infrastructure that the company has in place or will it require new infrastructure (hardware, additional software, and net ware); will the vendor

be in existence during the entire expected lifespan of the product; and what levels of support will they be able to offer in regard to upgrades, maintenance, and technical support? Other issues involve the ability of the software to support the company's existing and future communications protocols (e.g., X.12, XML), and the cost per user to run the software. All of these issues and many others need to be considered during the due-diligence period of vendor selection and should ultimately be captured in a legal contract.

Organizations need to determine their ultimate optimal business process requirements (business process re-engineering) prior to selecting the package, rather than letting the package completely dominate the selection process. While there might not be a complete one-to-one match between the organization's desired best-practice model and the process model provided by the system, this sequence of actions will allow the selection of a package that comes closest to the ideal, and adjustments may be subsequently made at the system and organizational levels.

Summary of positive issues

A wide variety of software packages is available, covering requirements for mass and niche markets. These include open source, freeware, and commercial software to which there is no code access; each type of vendor and code offers different solution options based upon the process and sourcing needs of the organizations.

Summary of potentially negative issues

While the range of vendors and solutions is wide, there is also a wide variety of solutions on the market. Some software is full of errors and bugs, some known, some unknown; and unfortunately price is not an indicator of quality as one would expect in, say, an automobile purchase. The quality of a software package needs to be examined from a process and technical perspective prior to purchase. This includes such testing as stress testing to determine the limits of an application to handle boundary problems, e.g., a sales-entry system that is capable of handling downloads of 1000 point-of-sale data entries per hour into its database might not be scalable and hence would be unsuitable for a hypermarket dealing with 10 000 entries per hour. The purchase of an application in a software package will come with a contract; however, all possible future issues can not be predicted in the contract. For example, in 1999 a contract for an accounting system would not have predicted the need to include provision for Sarbanes–Oxley; hence the question of who pays for this upgrade would not have been addressed, even a provision to ensure that the vendor make available, for a fee, upgrades to cover such process requirements might not have been included in the contract.

Reference

- N. Hollander (2000). *A Guide to Software Package Evaluation and Selection: The R²ISC Method* (New York, American Management Association).

Associated terminology: ERP, X.12, XML.

Spam

Foundation concepts: Email, Internet.
Definition: Unsolicited, unwanted, bulk commercial email.

Overview

Advertising is big business; direct mailing to consumers or potential consumers is very popular, as is demonstrated by the quantity of junk mail delivered daily. Bulk mailing of small cards costs 24¢ per item through the US Postal Service, and the cost is 39¢ per item for anything larger

than a postcard.[†] In addition to the cost of postage, there are also printing, assembly, and addressing costs, but still unsolicited direct-mail advertising abounds. Email is free. Hence spam.

The advantages to the advertiser are manifold. Complex typesetting and high-quality images may be used without cost; emails may be sent automatically to entire easily purchasable mailing lists without any human processing; and, most importantly of all, email is unregulated. In comparison with mail fraud, which is a well-understood crime, whose victims have little difficulty finding out how to report it, victims of electronic mail spamming have a more difficult time seeking redress, since the spammers work hard to circumvent the regulations and remain untraceable.

Although email messages seem to have return addresses, they do not really mean anything. Email protocols do not require a valid "from" address, and make no attempt to verify those provided. Once an email has reached the recipient, it is virtually impossible to determine where it really came from, so unscrupulous retailers feel free to say anything they want. Illegal and controlled substances, defective or nonexistent merchandise, stolen goods, everything is on offer.

Spam-prevention software generally attempts to flag unwanted email after it has been delivered, telling the recipient that they probably won't want to waste time reading it, but not preventing it from arriving. The effectiveness of even this technique is severely limited by the fact that spam can be detected only by reading and *understanding* the content of a message. The natural-language problem simply has not been solved: computers can not understand free-form human communication. Current techniques are barely more sophisticated than scanning the subject line for pre-chosen key sequences such as "Lotto Number Predictor" and the trade names of certain little blue pills. Deliberate misspellings or false headings easily defeat most systems.

Spam exists only because sending email is free. The decision that the cost of internet access for most subscribers should be charged as a monthly flat rate, with no usage-dependent costs, was arrived at mostly by default, and certainly favors the spam industry. If the origination of data-bearing transmissions actually cost something, some fractional amount, it would not impact the average subscriber: the total monthly bill would on average remain the same, it would just be calculated differently. Only those who transmit inordinate amounts would find significant increases in costs. There is no fundamental right to free long-distance telephone calls; it may seem surprising that internet traffic should be treated so differently, to the detriment of all users. The TCP/IP protocol family as it currently exists has no mechanism for accounting and charging, but far greater design challenges have been met before.

Business value proposition

The spam debate has positives and negatives depending upon whether you are a company that uses email as a direct-marketing channel or are an unhappy recipient. Email is a cheap mechanism for pushing a direct-mail campaign out to potential consumers. Recipients, however, find spam annoying and defending against spam to be a waste of corporate and individual resources.

Summary of positive issues

Spamming has led to the development of several laws to protect the consumer. The Controlling the Assault of Non-Solicited Pornography and Marketing (CAN-SPAM) Act of 2003 was developed to regulate commerce by imposing limitations and penalties on the transmission of unsolicited

[†] USPS pricing, obtained from www.usps.gov, September 2006.

commercial electronic mail via the internet. The act covers and attempts to prevent spamming and the avoidance of accountability by spammers. Its more important sections forbid the following actions:

- the unauthorized use of a protected computer to trasmit multiple commercial emails;
- the use of a protected computer to relay or retransmit multiple messages with the intent to mislead the recipients;
- the use of materially false header information in multiple commercial emails, and the distribution of such emails;
- registering five or more email accounts or two domain names using a false identity with the intent to transmit multiple commercial messages;
- the acquisition of email addresses by "harvesting" web sites without the consent of the site's operator;
- the false registration of an internet domain for the purpose of setting up a web site to capture email addresses; and
- sending commercial messages to randomly generated email addresses.

To protect the public and allow the spam filters to operate more effectively, the CAN-SPAM act stipulates requirements for the transmission of messages. The act requires that the header information be technically accurate: it must include a correct originating email address, domain name, or IP address. The act also requires that the "from" line in the email accurately identify the person sending the email, and the "header information" accurately represent the computer used for transmission of the email and not hide the fact that the email has been relayed through a series of machines. The header must also place a warning label on commercial email containing sexually oriented materials. The act prohibits deceptive subject headings and requires the inclusion of a return address or comparable mechanism in commercial electronic mail. The act also has a provision to establish a "do not email" registry that would allow recipients who object to receiving spam email to register, and requires that the sender must stop sending the email within ten days of the objection, but in a Federal Trade Commission report to the US Congress they stated that this could be counter-productive since spammers, possibly outside US jurisdiction, might use the list as a source of email addresses, and further that such a list "would raise serious security, privacy and enforcement difficulties."

Summary of potentially negative issues

The CAN-SPAM legislation has been criticized for having several gray areas and loopholes through which spammers can attempt to circumvent the law. Spammers continue to use a variety of mechanisms to place temporary or untraceable addresses in their email headers and even place just one line of "news" in the text to ensure that the message contains "newsworthy" content as the act demands. An additional problem is that a large volume of spam originates from nations beyond the jurisdiction of the CAN-SPAM law. The United States has not yet established a "National Do Not Email Registry."

References

- J. Zdziarski (2005). *Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification* (San Francisco, CA, No Starch Press).
- T. Muris, M. Thompson, O. Swindle, T. Leary, and P. Harbour (2004). *National Do Not Email Registry: A Report to Congress* (Washington, DC, Federal Trade Commission).

Associated terminology: Email, Internet, Internet protocol, Law cross-reference.

Spoofing

Foundation concept: Security.
Definition: A form of attack (attempted break-in or interception of data) in which one system pretends to be another.

Overview

Spoofing comes in many forms, the simplest of which, *IP-spoofing*, gives a good example of the whole spectrum. IP spoofing relies on the fact that computers on a network generally know each other by their *IP addresses*, but communications on a *Local area network* (LAN) use a totally different identification method, *MAC addresses*, instead. If a user (on computer A) wishes to make use of a password-protected server (on computer B) that has an IP address of 192.168.123.9, that user's computer will use the *address resolution protocol* (ARP) to find the equivalent MAC address. Unfortunately ARP is not a secure protocol. It works in a very simple way: the requesting computer broadcasts a "Who has IP 192.168.123.9?" query to every computer on the LAN. If everything is working properly, just one computer will respond with "I am 192.168.123.9, and my MAC is 11-22-11-44-33–11." The requesting computer notes that reply, and thenceforth sends all communications over the LAN directly to 11-22-11-44-33–11.

ARP is a very simple, efficient, and effective protocol, when everybody is on the same side. All it takes is for an attacker to momentarily make the real secure server inaccessible (perhaps just by unplugging its network cable for a fraction of a second, but there are many more sophisticated techniques), and transmit its own dishonest response to the ARP query, presenting its own MAC address as the translation for 192.168.123.9. ARP is so trusting that the invading system can transmit its dishonest response at any time, before or after the query is made, and the false answer will be noted and used by all. Of course, once computer A has confused computer C with computer B in its address-resolution tables, all transmissions intended for B will be sent directly to C instead. Computer C simply invites the user to log in exactly as computer B (the real server) would, records the user-name and password, and that's that.

Spoofing can also occur at the DNS (*Domain name service*) level. When a user is first making a connection to www.whatever.com, their computer will issue a DNS enquiry to find out the corresponding IP address. An attacker need only intercept the enquiry and provide their own false response, and the user will be tricked into connecting to the wrong server again.

There are many other levels on which spoofing can occur, since so many of the commonly used internet protocols are completely insecure. The best defense against spoofing is physical security, ensuring that no unauthorized systems can possibly be connected to the LAN. Of course, it is incumbent upon the administrators of other levels of the network, and ISPs, to provide the same security for their equipment.

The adoption of secure encryption-based protocols instead of the traditional insecure ones would make spoofing virtually impossible, but replacing the standard internet protocols either requires agreement and cooperation from the entire internet community, or results in a locally secure network that can not communicate with the rest of the insecure internet.

Business value proposition

The use of spoofing techniques to fool a secure source into giving access rights or other privileges to an unauthorized computer is a well-known and easily tried technique used by hackers to gain valuable information, should a network be left undefended. Therefore it is vital that the CIO act through the company's chief security officer and network administrator to

ensure that all adequate protections are in place.

Defending against spoofing is best achieved through physical security measures, ensuring that no unauthorized computer can access the network, and this policy should be enforced at all other network levels, including the ISP. This requires that organizations use ISPs and service providers that take spoofing seriously and actively enforce their policies. Concern in this area may lead to the decision to use only private networks and to adopt secure encryption-based protocols rather than relying upon the insecure ones.

Summary of positive issues

Most spoofing techniques are well known and may be defended against through the use of private networks, encryption, and other techniques.

Summary of potentially negative issues

New spoofing techniques are always being developed and resources need to be deployed to keep track of threat levels and address any newly revealed security weaknesses.

Reference

- B. Schneier (2004). "Customers, passwords, and web sites," *IEEE Security and Privacy*, Volume 2, No. 4.

Associated terminology: Phishing, Virus, Trojan horse, Worm.

Spyware

Foundation concept: Security.

Definition: Illicit and usually illegal software that spies on a computer's stored data and its users' activities, reporting them through an internet connection or an autodialing modem to a third party.

Overview

Market research and advertising consume an enormous proportion of many corporations' budgets; however, some companies simply lack the funds to develop sophisticated market research and advertising campaigns. A system that would provide almost unlimited opportunities for market research and direct-to-consumer advertising would be very tempting indeed. That is exactly what *Spyware* is.

Simply visiting a web site once is enough to cause software from that site to be downloaded, installed on your own computer, and executed, if your web browser does not have sufficient security settings selected. Software can similarly be embedded in emails and automatically installed if the email client application is not secure. Very commonly, spyware takes the form of a *Trojan horse*, some piece of free software that is claimed to perform some useful task is made available for download through the internet. Many users will happily download it, install it, and try it out. Maybe it does perform its advertised task, and maybe it doesn't, but, if it is a spyware delivery vector, it will continue to run after it appears to have stopped, and will have added itself to the system's start-up list, so that it will automatically run every time the computer is started.

Once it is running, spyware has the unobstructed use of the entire computer. It can be designed to search out certain files (address books, password files, etc.), or to look for any file containing certain kinds of data (account numbers, PINs, etc.) and send the contents to the spyware's author. It can monitor the keyboard and record everything that is typed; it can record every application that is run, and every file that is accessed; it can record every web page that is viewed, and every email that is sent or received. Everything the computer does can be recorded and transmitted. It could even potentially turn on a web-cam and capture real-time pictures of the user in their home or office.

Traditionally, the term spyware is limited to software that passively spies, as detailed

above. Software is not limited by the term used to describe it, and spyware will commonly play a more active role. Spyware can be written to redirect web searches, so that the operators can insert their own preferred sites instead of the results provided by the genuine search engine. It can redirect simple web accesses so that the operators' advertisements are shown instead of the real pages. The most common non-passive act is to "pop up" unsolicited advertisements.

In many cases, spyware is illegal, but, in the United States, legislation to protect private parties (both individuals and corporations) is very weak at the federal level (government computers and those of financial institutions are protected, of course), and left to state legislatures. State laws are not at all uniform, and, since most use of spyware is an interstate attack, state laws can be difficult to enforce. Many spyware operators manage to convince themselves that their actions are neither legally nor morally wrong, and are even willing to believe that their victims are really beneficiaries.

Spyware is mainly a plague on users of personal computer systems. This is partly because spyware requires some effort to create, so that effort is directed toward the systems that most people have. It is also partly because of the large numbers of security flaws discovered in popular operating systems and third-party software made for them. The only way to maintain privacy is to avoid spyware, and the only way to avoid spyware is to run a secure computer system, in particular never running software that didn't originate from a completely trusted source. As with viruses, there are a few anti-spyware applications available, and in this case some of the best are also free.

Business value proposition

For corporations and individuals, spyware is generally an annoyance that is ignored until it builds to a critical level and prevents proper system functions. Much depends upon the type of spyware that is resident upon a system. Typically spyware attaches itself to a computer through an access to a web site or through an email message, and then collects information on the computer or its user's activities. This information is then transmitted over an internet connection to a remote user. Clearly, in the worst instances spyware can run undetected for long periods of time, transmitting industrial secrets to a third party, or set a web-cam in action, monitoring an individual's activities in the "privacy" of their home. However, not all spyware is put on a system from an external source; spyware can be placed on a system by an employer to monitor employees' compliance with company policies, or to check against access to unauthorized sites. Parents may wish to place spyware on their children's computer for similar reasons.

The best way to avoid spyware is by not being connected to the internet; however, this is not always practical and the next stage is to install a *Firewall* and software to filter out problem files. The computers themselves can also have their security levels raised to forbid the installation or running of web-resident software. Finally, there are many commercial vendors who provide anti-spyware software (some for free) and these act to sweep all aspects of a computer's memory, flagging and then removing undesired spyware and adware. Spyware itself evolves continually and thus it is necessary for users to update their software's library of possible spyware sources and categories.

Summary of positive issues

Spyware can be used to actively monitor any computer. Companies may wish to monitor the online activity of an employee, whereas a parent may wish to monitor their child's online behavior. Anti-spyware software is available. Certain categories of spyware activity are illegal in some

jurisdictions. In the UK, all software installed on a computer without the owner's consent is illegal. In the US many states have similar laws, but there is no federal equivalent.

Summary of potentially negative issues

Spyware can cause serious damage to an organization by stealing intellectual property, passwords, or other corporate data. Lesser damage is caused by its occupation of valuable network bandwidth and slowing of infected computer systems, but this can become critical if left untreated for too long. Spyware can be created to monitor the personal behavior of a computer user, for example their online shopping activity and their web-surfing habits, or even to turn on computer equipment such as web-cams. Legal recourse against spyware is limited in many jurisdictions to certain categories of software and intrusion. Spyware may be impossible to detect without the aid of special software.

Reference

- E. Schultz (2003). "Pandora's box: spyware, adware, autoexecution, and NGSCB," *Computers and Security*, Volume 22, No. 5.

Associated terminology: Advertising, Virus, Trojan horse, Phishing.

Storage

Definition: Memory, disks, and other media used to retain data and applications both in the short and in the long term.

Overview

Memory or storage is one of the *sine quibus non* of any computer system. Unlike with humans, it is not possible for a computer with poor memory retention to get by. The *central processing unit* (CPU) of a computer, its brain in anthropomorphic terms, has no capacity for any sustained activity; hundreds of millions of times per second, it must access its memory to find out what it is supposed to be doing next. Without memory, it would grind to a halt in a few nanoseconds.

Computer storage is usually divided into two distinct categories, *Primary storage* and *Secondary storage*. Primary storage, often just called *Memory*, is the kind that is essential for the nanosecond-to-nanosecond operation of the processor. Every instruction to be obeyed by the computer, and every piece of data to be used, must be present in primary storage. This means, of course, that a computer must have as much primary storage as possible, but the importance of size is nothing when compared with the importance of speed and reliability. The processing speed of a computer is completely bound by the speed (or *Access time*) of its memory.

There is an essential balance to be struck. The faster memory is, the more it costs, and the more memory costs, the less of it a computer can have. Additionally, faster memory consumes more electrical power, and results in more heat that must be dissipated from the circuits, which is a major problem in modern computer design. The generally used solution is a complex compromise. A typical modern computer has a large amount of reasonably fast memory known as *Main memory* or "*RAM*," usually in the hundreds of millions of bytes, plus a much smaller amount of exceptionally fast memory, known as the *Cache*. Whole applications and data sets are loaded into main memory as usual, but, as an application runs, the portions of that data that are in immediate use are copied into the cache. This is a very complicated dynamic process performed by the CPU, but it is responsible for significant improvements in processor speed at minimal cost.

Another economic factor is the unsurprising fact that it is much cheaper to make *Dynamic RAM*, memory that can retain

information for just a few thousandths of a second, than it is to make *Static RAM*, which retains information for unlimited periods. The difference is so great that all modern computers use almost exclusively dynamic RAM, and are equipped with extra circuitry required to *Refresh*, or remind every memory cell of what it is supposed to be storing, hundreds of times per second automatically. So much research and development has been devoted to improving the performance of dynamic RAM that the overall effect is a benefit. Static RAM has been relegated to a very small unit that runs on battery power to keep essential system settings when a computer is turned off.

Another unfortunate economic aspect is that the only affordable memory is *Volatile*. That means that it works only when the power is on. Up until the mid 1970s, when all computer memory was expensive, the main memory technology in use was magnetic: vast arrays of minute magnetizable doughnuts, called *Cores*, that each stored one single bit in a magnetic field. These cores were *Non-volatile*; a computer could be turned off for a week, and still retain everything. Unfortunately, core memory takes up a lot of space, is much slower than semiconductor memory, and is vastly more expensive. The only part of core memory still surviving is the name; many still call main memory "the core," and the extensive status report produced when an application crashes is called a *Core dump*.

One cheap form of non-volatile memory is available; it is known as ROM, or *Read-only memory*. As the name suggests, ROM provides only half of the functionality normally required of computer memory: its content can be read (i.e., looked at), but can not be written (i.e., modified), so it is suitable only for data that absolutely never changes. Even essential basic applications and operating systems can not reasonably be stored in ROM because it would make upgrades impossible. ROM comes in many varieties. The basic kind must be programmed with its permanent content at the factory. PROM (*Programmable ROM*) comes from the factory blank, and can be programmed (once only) using special equipment. Another variety, EPROM (*Erasable programmable ROM*) can actually be erased and reprogrammed (erasure is either by exposure to ultraviolet light, or electronically), but erasure is a very slow process, and EPROM does not provide a viable form of general-purpose non-volatile memory. ROM is used primarily for controlling embedded processors and for storing the permanent system-start-up procedures (*BIOS*) in a modern computer.

The term "*RAM*," which is now used to mean general-purpose memory, is in fact an acronym for *Random-access memory*. Random access means that the various items stored in the memory may be accessed in any order at any time, rather than *Serially*. A good analog is the difference between watching a film on a DVD and on a video tape: DVDs are random access, you can skip to any point at any time; video tapes are serial, the only way to get to a point half way through the film is to wind through half of the tape. There is no serial-memory technology in current use, and the meaning of *RAM* has changed slightly. Technically speaking, *ROM* is a form of *RAM*; its contents may be accessed in random order. In modern usage, however, *RAM* means strictly normal general-purpose memory.

Because non-volatile RAM is prohibitively expensive, and even the usual volatile RAM is by no means cheap, computers also need secondary storage. In the economic trade-offs of computer design, primary storage is chosen to be as fast as possible without becoming too small to use; secondary storage is chosen to be as large as possible without becoming too slow to use. Secondary storage currently uses totally different technology, almost invariably the *Hard-disk drive*. Disk drives use magnetic fields to store data, and so are non-volatile, retaining data for many years without use. The

magnetic fields are stored in concentric circular tracks on a rapidly spinning disk (hence the name). Typically, random access to an item of data stored on disk takes of the order of a million times longer than random access to a data item in primary storage, but capacities tend to be approximately one thousand times higher for similar prices. The significantly greater capacities and non-volatility of disks make the combination of semiconductor RAM for primary storage and disks for secondary storage a universal choice.

Business value proposition

The selection and use of storage in a corporate setting requires that a business case be developed for the whole system, taking into account the processes that are going to run upon it. In a system for a person whose only requirement is to use a word processor, memory speed and size will typically not be the bottleneck, but rather the person's typing speed. However, in the case of a file server that is used by corporate users, an assessment of memory requirements must consider issues of performance, loads, and network demands. Adding lots of memory to a corporate server that has a low-performance processor could still greatly increase performance.

RAM is available in a wide variety of types, each of which is available with a wide array of performance options. These choices facilitate the acquisition of a specific memory component for a specific purpose (e.g., cache for main memory), and allow systems to have their memory upgraded, *potentially* meeting new requirements without the user having to purchase a completely new system.

Commercial and personal computer users usually never change the ROM configurations of their systems, since these typically store system-specific applications, but newer systems use *Flash ROM*, which does allow the BIOS to be updated without any special equipment needs.

Technologists can, however, place their software into a hardware memory device that is non-volatile in nature if desired. Various devices are available that allow developers to burn their software into a PROM, test the program in the PROM, and move the PROM onto production memory devices.

Secondary storage is available in a wide variety of configurations and needs to be considered as part of a company's *Information lifecycle* (ILC) and the management of the company's data. There are many companies offering data-storage services that can be used as outsourcing partners.

Summary of positive issues

A wide range of devices and specifications is available for both primary and secondary storage. The technology is well understood and supported by vendors. Software and hardware tools are available to help users to analyze and optimize their memory. New forms of memory are continually evolving (e.g., USB flash-memory devices). Primary storage is available in a wide variety of configurations.

Summary of potentially negative issues

The wide variety of memory types requires that the correct memory be selected for each situation. Adding memory to a computer system might not make the system more effective or faster since the bottleneck may lie elsewhere in the system. Some memory specifications and types lose their manufacturer's support as they age and are superseded by newer specifications. While primary memory can last for ever, barring physical harm, secondary memory has a shorter lifespan because it contains moving parts, and an assessment of the condition of the memory needs to be undertaken regularly, so that the appropriate upgrades, maintenance, and adjustments can be made.

Associated terminology: Memory, Disk, Optical storage, Backup, Memory and disk size.

Structured design methodologies

Foundation concept: Software development lifecycle.
Definition: Structured design methodologies are a set of techniques available to systems analysts and programmers to assist them in producing system designs and programs with higher levels of validation and verification than would otherwise be attainable.

Overview

Structured design methods started to appear in the late 1960s, when the number of systems under development started to increase dramatically and the quality of those systems, in terms of the code matching the specification, was generally poor. In addressing this, researchers such as Dijkstra, Wirth, Baker, Knuth, Stevens, Myers, and Constantine focused upon developing structuring techniques to help developers and implementers produce better systems. Some of the researchers concentrated on design issues and some upon technical and implementation issues, and eventually, by the 1980s, a series of methodologies and techniques for the creation of complex high-quality systems had been developed.

The term *Structured design* was originally used by Stevens, Myers, and Constantine in their book *Reliable Software through Composite Design*, which was published in 1974 and in essence an extension of what IBM had termed the *Hierarchical input process output documentation technique* or HIPO for short. The structured design techniques use box-and-line notation in order to present the design diagrammatically, with boxes of certain shapes representing a type of object such as a *Module* or a *Predefined module*, the lines linking boxes together in a tree structure with a notational marker associated with each link between the modules stating what data is passed between modules.

It should be noted that there is no one universal diagrammatic structure or notation; many different notations have been developed since the 1970s, some as open standards such as Wirth's *Step-Wise Refinement*, and some associated with a particular company or designer such as *Jackson Structured Programming* and the *Yourdon Design Method*. All of the methods do tend to have some shared properties, and involve structures that are primarily based upon the hierarchical decomposition of functional modules into smaller modules with specifications of the data passing between them. The modules are usually drawn or marked in such a way as to denote that a sequence of activates occurs in that module (step one, then step two, then step three), that a selection process occurs (perform process one or perform process two), or that an iterative process occurs (perform process one until condition two occurs).

Various methodologies treat the constructs with different rules to avoid conflict in the code (e.g., *Jackson Structured Design* does not permit sequence, selection, and iteration to be at the same level in a tree; a sequence of activities that themselves can be decomposed into further activities is permitted, but no one node may be directly decomposed into a mixture of the different kinds of sub-nodes). Such rules are intended to ensure that the final *leaf* nodes in the design tree represent program statements or activities that ultimately can be translated into program code with good design characteristics.

Structured design was originally aimed at the development of transaction-processing systems in languages such as Cobol, but has evolved in various directions since the 1970s and led to methodologies such as *Entity-relationship diagrams* (ERDs) and *Data-flow diagramming* (DFD). These and other structured methods have also been built into the software lifecycle models such as the *Waterfall model*, which is used

to develop a concept into a program, and encourages the use of structured methods at each stage. The structured approach is also employed in commercial methodologies such as *RAD*. The structured approach continues to be used in various forms and hybrids and has evolved into methods such as UML to accommodate the current *Object-oriented* paradigm. Structured models are generally not considered rigorous enough to develop mission-critical systems that may, for example, involve potential risk to human life; for such systems a more mathematically rigorous approach to design, known as *Formal methods*, was developed.

Business value proposition

The use of structured design techniques leads to higher-quality, more maintainable, and better-documented programs and systems than would be the case if no unifying design approach were taken. The adoption of structured methods allows companies to use a consistent approach to developing their systems, allowing consistency in training and more efficient planning and scheduling, recruitment, and management. The structured approach allows for the adoption of commercial tools and packages to support this style of development as well as the adoption of management practices such as RAD to speed up development, and cost modeling techniques such as COCOMO II to help determine system costs and effort requirements. The methods are well known and supported by consultants and software tool vendors.

Summary of positive issues

The structured approach to systems development is well established, with a long history and extensive literature. A wide variety of software tools is available to support structured methods and the associated management techniques that accompany this approach to systems design.

Summary of potentially negative issues

Structured design was developed in the 1970s to aid in the design of transaction processing systems in Cobol. Early methodologies may be less suitable to the development of object-oriented program designs and care needs to be taken in methodology selection. Structured methods tend to be considered "less rigorous" in nature since they don't involve the mathematical formalism of formal methods; as a consequence they may be considered unsuitable for systems requiring absolute precision in the development process.

References

- E. Yourdon (1979). *Classics in Software Engineering* (New York, Yourdon Press).
- G. Myers (1974). *Reliable Software through Composite Design* (New York, Mason and Lipscomb Publishers).

Associated terminology: Formal methods, UML, RAD, Object-oriented, Cobol, ERD, DFD, Waterfall model.

T-Carrier

Foundation concepts: Network, Bandwidth.
Definition: T-Carrier is the generic name for digital high-speed data-transmission lines more specifically known in the United States as T-1, T-2, etc.

Overview

The basis of the T-Carrier system is the digital *Channel*, designated DS0 or T-0. A single DS0 channel has a capacity or *Bandwidth* of 64 kbps (64 000 bits per second), which is the amount of carrying capacity that Bell Labs originally defined as necessary to carry a digitized voice signal over their network. Originally, the name T-1 was used for a digital transmission channel with a bandwidth of 1544 kbps that carried 24 separate DS0 channels together with one 8 kbps channel for system-related information. Now the technology may be slightly different, and T-1 has become a less specific name for any 1544 kbps digital carrier. Similar names (T-2, T-3, etc.) are used for higher-bandwidth carriers:

- a T-1 line is equivalent to 24 channels or 1.544 Mbps,
- a T-2 line is equivalent to 96 channels or 6.312 Mbps,
- a T-3 line is equivalent to 672 channels or 44.74 Mbps,
- a T-4 line is equivalent to 4032 channels or 274.2 Mbps, and
- a T-5 line is equivalent to 5760 channels or 400.3 Mbps.

although it is not common to hear of anything other than T-1 and T-3. In Japan, exactly the same system is used, except with the letter J instead of T. Originally T-lines were made of copper wire, and the designation FT was used for fiber optics, but now the distinction is often ignored.

In Europe, a slightly different system is used, still based on 64 kbps DS0 channels (called E-0), but in multiples of 32 rather than 24 (giving a bandwidth one third higher than that of the corresponding T-channel), and using the letter E instead of T. So an E-3 carrier, for example, would have 33% more bandwidth than a T-3 carrier.

Business value proposition

Organizations can purchase capacity from common carriers in terms of a leased T-*x* carrier line. For example, a company can lease a whole T-1 line, or, if a T-1 line provides more bandwidth than required, a smaller number of DS0 channels (known as a *Fractional T-1* subscription) may be available at a lower price.

It is important to note that the speed of downloads or uploads through a network depends not only upon the bandwidth provided by the ISP and common-carrier connection, but also upon the nature of the network that it is connected to. Thus, for speed-sensitive environments, it is worth testing the actual speed of transfers.

Summary of positive issues

ISPs and common carriers provide a variety of bandwidth capacities. The European standards are compatible with the North American and Japanese standards.

Summary of potentially negative issues

Organizations need to check the bandwidth that they are actually achieving given the nature of the network through which they are connected.

Reference

- The International Telecommunications Union (http://www.itu.int).

Associated terminology: ISP.

TCP/IP (Transport Control Protocol for

Foundation concepts: Network, Protocol.
Definition: The preferred protocol for reliable network communication.

Overview

TCP/IP is often thought of as a single thing, but TCP and IP are separate protocols with separate purposes. IP (*Internet protocol*, q.v.) is the logical mechanism used to deliver data over a network; it provides an addressing and adaptive-routing system that allows one computer to communicate with another without knowing where it is or how to find it. IP does not provide any reliability of service: most transmissions arrive at their destination, but not all. IP does not depend upon any other particular protocol, it often uses ethernet, but does not have to; in particular, it is not bound to TCP in any way.

TCP (*Transport Control Protocol*) is one of the higher-level systems that are built on top of IP to provide additional functionality. TCP does require IP, but not vice versa. TCP relies upon IP to attempt to deliver data correctly, but adds its own procedures for ensuring reliability.

A networked application that requires reliable communications, such as an email server, *could* use IP to send all of its data. In that case, it would have to be carefully designed to take account of the fact that some transmissions will not arrive, some portions of the data may arrive in the wrong order, and some may even arrive twice. The client and server would have to exchange acknowledgements for every piece of data, establish time-outs so that lost data will eventually be re-sent, and provide data identification so that duplicated and disordered data may be corrected.

TCP provides all of that functionality automatically. The same application using TCP plus IP instead of IP alone would have no extra work to do beyond simply sending the required data. TCP provides applications with thoroughly tried and tested procedures for all the essentials of network communications, relieving developers of one of the most difficult and error-prone tasks.

TCP is quite a complex protocol, and is not always the most appropriate choice for all applications. The need to transmit acknowledgements for data, and acknowledgements for acknowledgements, puts a significant load on network capacity. In cases in which responsiveness is more important (such as network telephony) or reliability is not required (such as a network time service that transmits the time of day every minute: it would not matter if a few transmissions went astray), UDP (User Datagram Protocol) is a popular and efficient alternative.

The division of responsibility between IP (routing) and TCP (delivery) is part of one of the most important principles of systems design: *Divide and conquer*. Two separate systems, individually testable and with their own non-overlapping responsibilities, will be much easier to develop reliably than will a single system trying to perform both tasks. For network programming, divide and conquer is usually taken much further, with systems divided into many layers. The most popular view is the OSI (Open Systems Interconnection) *Seven-layer model* (see *OSI* for details), but the more traditional four-layer model is often preferred. The layers are as follows.

Layer 1, “link”: this describes the physical connections (cables, connectors, etc.) and how data is sent directly between two network participants that have a direct physical link. Example: ethernet.

Layer 2, “network”: this describes how data moves around a network as a whole, providing addressing and routing procedures. Example: IP.

Layer 3, “transport”: this describes the flow of data between one application and another, providing reliability to whatever degree is desired. Examples: TCP and UDP. Whereas IP is concerned with delivering data to the right computer, TCP and UDP are concerned

with directing it to the right application running on that computer.

Layer 4, "application": this is the "useful" layer, performing the high-level tasks that are actually wanted, such as delivering email or remotely controlling equipment. The other layers are really just services, whose sole purposes are to make this fourth layer possible.

Business value proposition

TCP in conjunction with IP is the most widely known protocol for general network communications. However, it should not be forgotten that there are alternatives. When deciding upon which protocol to choose, one must develop a business case that examines the future requirements of the systems under consideration. This can be done through the creation and evaluation of a set of performance indicators (such as reliability) and then systems metrics (such as performance requirements) determine which protocol would be the most suitable. The business case will also consider the protocol requirements of other systems that the network would interact with, including those beyond the corporate boundary. Many organizations will also have technical staff capable of creating a proprietary protocol that exactly fits their specific needs; a proprietary layer-3 (transport) protocol may certainly be a viable solution, since it would ride upon IP, the *lingua franca* of the internet; a proprietary layer-2 (network) protocol is probably impractical, because it would not be recognized by existing network equipment.

Summary of positive issues

TCP is a well-known, highly reliable, and established protocol.

Summary of potentially negative issues

TCP is a complex protocol, which places a higher demand on networks and thus may require a higher network bandwidth capability than less convenient alternatives would need.

Reference

- W. R. Stevens (1994). *TCP/IP Illustrated* (New York, Addison-Wesley).

Associated terminology: Internet protocol.

Telnet

Foundation concepts: Client–server, Operating system.

Definition: A network application that gives remote access to any software that provides a command-line interface and is itself network enabled.

Overview

Before graphical user interfaces and windowing systems became affordable and popular, the regular way to control a computer was by typing verbal commands on a keyboard and reading a typed response. This kind of interaction is still familiar to anyone who uses Unix or "the DOS prompt," and is often still the preferred mode of operation for programmers and the technically minded. There is only so much information that can be conveyed by clicking and dragging icons, and, when more is needed, the use of complex dialogs to provide the extra information can be very tedious and inefficient.

The use of a *Command-line interface*, through which users type commands that may be as simple or as complex as is desired, can provide a great increase in efficiency. The only cost is that users must learn the language of the interface. Dialog windows do at least ask for specific information in specific places, so a user need only understand the questions in order to use them properly. When faced with a command-line prompt, there is generally no information on what the system will accept, and a user can not reasonably guess at the syntax and hope for the best. Use

of a command-line interface requires some preparation and even expertise.

Some operating systems have a command-line interface as their primary means of control; Unix (and its variants, Linux, BSD, etc.) is a prime example. Control of the system and interaction with applications are both conducted through typed commands (although windowing systems are of course available). Other operating systems (for example windows) are primarily controlled through a graphical user interface, but individual applications may still provide their own command-line interfaces for convenience.

Telnet is a simple network application that provides remote access to any system or application that provides a network-enabled command-line interface. *Network enabled* is a key requirement. The system or application must be designed not just to accept commands typed on a keyboard, but also to act as a *Server*, willing to accept TCP/IP connections over the internet, and treat transmissions received over that connection as though they had been typed at the local keyboard. This is not at all difficult for an application to do, but is still essential. If the operating system itself provides a command-line interface, then that interface is normally inherited by all applications that run, so no extra programming is required.

The Telnet application acts as a *Client*, connecting to any server as directed by the user, then simply acting as a communications intermediary. Anything typed at the keyboard is transmitted directly to the server's command-line interface; any output produced by the server is displayed immediately for the user to see. Telnet provides a simple, cheap, and uniform interface that allows systems and applications to be controlled from any networked location.

Pure Telnet is a very insecure system. If legitimate users can connect to a system using Telnet, then so can anyone else. Of course, password protection will prevent unauthorized access if implemented correctly, but typing passwords over network connections can be very unsafe. When the legitimate user types their password, it must be transmitted to the server for verification. Anyone with a tap on the network (e.g., a *Packet sniffer*) can capture transmissions and learn the password. As a consequence, Telnet should be used only over a secure connection, protected with SSL (*Secure-sockets layer*), SSH (*Secure shell*) or a functionally equivalent encryption scheme. Secure Telnet clients are widely available.

Telnet is also known as the Hacker's Friend, since it allows human users to communicate directly with automated servers devoted to some network task. For example, it is very easy for someone familiar with the SMTP protocol (the language of outgoing email servers) to connect directly to a server and send email messages with fraudulent return addresses and origination dates. This is, of course, really a problem with unsecured servers rather than with Telnet itself; if Telnet did not exist, someone would create it.

Business value proposition

The value of Telnet is that it provides a simple command-line means of accessing a remote system via a network connection. This enables users to connect to remote services as desired over the internet. A positive aspect of the Telnet approach to systems access is that network managers can remotely log into their system and perform any system function as desired. Thus, for example, if an oil company has a Unix system on an oil rig, the network manager can restart processes and perform maintenance from his or her office and does not have to board a helicopter and fly to the middle of the ocean, which would probably be the case for a systems manager trying to fix a non-"reboot" problem via an operating system without this access feature.

Summary of positive issues

Telnet provides a tried, trusted, and convenient method for accessing remote computers over a network. Applications to add varying levels of security to the system are readily available.

Summary of potentially negative issues

Pure Telnet is a vulnerable access method and needs to be reinforced with methods such as SSL to encrypt the messages. Telnet connects to line-command operating systems and applications and these require instruction and training of users since they are not as intuitive as the drag, drop, point, and click operating systems.

References

- W. R. Stevens (1994). *TCP/IP Illustrated* (New York, Addison-Wesley).
- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Protocol, Internet.

Thirty-two bit

Overview

This is closely related to the discussion of memory and disk size (q.v.). To a computer, a *Bit* is the same as a *Digit* is to a human. If you had a desk calculator that could work only with three-digit numbers, it would be very inefficient to use: it could calculate 5×7 perfectly well, but if you needed to know 1215×49 there would be trouble. The calculator would still be better than nothing, you could use it to calculate 12×49 and 15×49, and then combine the results, but it would be a lot of work. A calculator that could handle up to eight-digit numbers would make everything so much easier.

The same is true with computers. An eight-bit number is restricted to the range -128 to $+127$: more or less two-digit numbers in human terms. A 16-bit number is restricted to the range $-32\,768$ to $+32\,767$: roughly four human digits. A 32-bit number corresponds to approximately nine human digits, and a 64-bit number has up to 19 digits in human terms.

An eight-bit computer has the hardware required to work on eight-bit numbers directly. It can still work on larger numbers, but has to perform a lot of extra work to do so. In general, a 32-bit computer can perform a calculation on a nine-digit number in the same time as a corresponding 8-bit computer could perform the same calculation on a two-digit number. More bits means more processing power.

The first microprocessors were 4-bit, and never found their way into general-purpose computers. The first desktop computers, in the 1970s, were 8-bit. The first commercial PCs (the IBM PC AT and XT) were 16-bit. Currently, 32-bit processors are in the vast majority of computers (the Pentium and all of its relatives are 32-bit), but 64-bit processors are gradually gaining ground.

The number of bits that a processor has can be measured in a number of ways. The most common is the width of the *Arithmetic logic unit* (ALU), which determines the magnitude of the numeric values that can be processed in a single operation. Another common measure is the width of the *Data bus*, which determines how much data can be retrieved from memory in a single operation, and yet another is the width of the *Address bus*, which determines how much memory can be directly accessed. These three measures all have a strong tendency to increase as technology progresses, but they do not increase uniformly. Most desktop computers today have a 32-bit integer ALU, an 80-bit floating-point ALU, a 64-bit data bus, and a 36-bit address bus. The exact meaning of the phrase "thirty-two bit" must be determined from its context.

Software is specific to a particular kind of processor. Most programs may be designed

to be *Platform-independent*, which means that they could potentially be run on any computer, but the application itself needs to be recompiled (reconstructed) for each kind of processor, and system-critical software (most especially the operating system) is often designed to take advantage of specific features of specific processors. Software created for a 16-bit processor *may* still work on a 32-bit processor, but it will not be able to take full advantage of the new processor's capabilities. System software created for a 16-bit processor often does not work at all on a 32-bit processor.

Thus, there is 16-bit, 32-bit, and 64-bit software. The number of bits is not really an aspect of the functionality of the application, but an indicator of the kind of processor it works (best) on. As hardware is upgraded, it is often necessary to upgrade software too. Conversely, sometimes a software upgrade may unexpectedly require a hardware upgrade. The best-known example is probably from the history of Windows. Windows 3.11 was a very popular 16-bit operating system. It had been designed to be compatible with earlier 16-bit processors, and could not properly support 32-bit software or make effective use of the 32-bit microprocessors (such as the Intel 80486) on which it most commonly ran. It was replaced primarily by Windows 95, which is a 32-bit operating system, and could make much better use of the processor's power, but a lot of software designed for 16-bit systems became inoperable when it was installed.

Business value proposition

An aspect of technology that continues to evolve is the number of bits a computer uses internally to perform its computations and other operations. Mainframe-style computers, for which keeping the price low was not such a vital design decision, traditionally had large bit sizes, such as 36 or 60. Microprocessor technology, which is far more price-driven, has evolved from computers that happily used 4 bits (although not so happily for their programmers), to computers that used 8 bits, then 16 bits, followed by 32 bits and 64 bits. A larger bit size allows computers to quickly access larger amounts of memory, produce more realistic-looking graphics, and perform computations involving larger or more accurate numbers with greater efficiency.

The consequence for organizations relates to the procurement decisions surrounding both hardware and the software. As the hardware of computers changes, software vendors produce new packages or new versions of their existing software, which might not be compatible with older hardware. Frequently, using the most up-to-date software requires purchasing the most up-to-date hardware, and this may be a significant expense. When legal requirements change, the high cost of an upgrade may be unavoidable.

This lifecycle forces users to upgrade their systems more frequently than they may have expected, and users who may be happy with their existing systems have to endure the expense and upheaval of an unwanted upgrade. The upgrade requirements are compounded by the fact that an upgrade to one aspect of the system may require an upgrade to the whole system to ensure that components continue to interoperate with each other. Clearly, there is also a danger in being first mover to a new platform because systems bugs and incompatibility problems might not have been corrected or even discovered, so any move to a new platform needs to be carefully considered.

Summary of positive issues

Larger bit sizes enable computers to perform faster computations, data transfer, and processing, to access more memory, and to produce better graphical displays.

Summary of potentially negative issues

The transition from one generation of computer to another may require software vendors to re-write or adapt their system to meet the new hardware architecture. This action frequently requires users to upgrade to the new hardware platform and the new version of the software at additional cost.

Associated terminology: Bit, Bus.

Transaction processing system (TPS)

Foundation concept: Database.

Definition: A transaction processing system is an information system whose principal intent is to process complete transactions involving "goods, service or money" (www.tpc.org). For database systems, a transaction is a set of operations that must be performed as one, in order to ensure consistency.

Overview

The term *Transaction processing* (TP) can be traced back to the early *Data*-processing (DP) era of information systems (approximately 1969–1979) when the primary activity that occurred in the *Data center* was the repetitive processing of data through programs that resided upon a mainframe computer operating in batch mode. The organizational view was that the TP system needed to work at maximum operating capacity in order to achieve an economic payback on the large investments that the mainframe environments required.

Since the 1980s, as systems became more integrated and moved from batch processing to online processing, transaction-processing systems came to be known as *Online transaction-processing* (OLTP) systems. The early OLTP systems differed from their batch predecessors in that they allowed online (interactive) processing and retrieval of data as well as reporting capabilities.

OLTP systems have since become embedded within larger systems such as ERP systems, and the functions that were once stand-alone have become "modules" in these larger systems. They still perform their transaction-based functions but are fully integrated with the other modules of the systems. However, it should be noted that extremely high-volume transaction processing of such items as credit card bills, for which millions of items of data need to be passed through the same program, is still performed as a dedicated process on mainframes designed for TP at high processing speeds.

In database systems, a transaction is defined to be a set of operations that must be completed as a whole or not at all. For example, when funds are transferred from one account to another, the debiting of one account and the crediting of the other have to be performed as separate operations on the database, but must be treated as a single transaction, so that, if there is a system failure, it will be certain that either the transaction happened or it didn't: there can be no half-way result. The design of efficient, effective, and guaranteed *Commit* and *Rollback* operations that ensure this behavior in transactions is a major challenge in the implementation of database systems.

One organization that attempts to benchmark the performance abilities of computer systems on standard tasks is the Transaction Processing Performance Council (www.tpc.org), whose aim is "to define transaction processing and database benchmarks and to disseminate objective, verifiable TPC performance data to the industry." This organization publishes the results of its benchmarking studies for consumption by the computing industry and consumers.

Business value proposition

Transaction processing systems facilitate optimal resource utilization and the efficient processing of large blocks of data (e.g., a payroll module of an ERP can be run at night when system usage is low).

Summary of positive issues

Transaction processing systems provide a cost-effective mechanism for the processing of large amounts of functionally similar transactions. Vendors and outsourcers are available to support TP.

Summary of potentially negative issues

Transaction processing systems, depending upon the scale of data passing through the system, may require significant dedicated resources.

References

- P. Bernstein and E. Newcomer (1997). *Principles of Transaction Processing* (San Francisco, CA, Morgan Kaufmann).
- Transaction Processing Performance Council, Presidio of San Francisco, Building 572B, Ruger St., San Francisco, CA 94129-0920, USA.

Associated terminology: Batch processing, ERP.

Trojan horse

Foundation concept: Security.

Definition: A piece of software, disguised as something desirable, but which is actually maliciously destructive.

Overview

According to Homer (the Greek, not the Simpson), some time around 3000 years ago, the Greeks laid siege to the city of Troy (now known as Hissarlik) in western Turkey, in order to retrieve Helen, the abducted/escaped wife of the King of Sparta. The siege seemed to be heading for failure, until the Greeks devised a new plan. They built a giant wooden horse, which the Trojans would believe to be a tribute from a humbled enemy. The horse was secretly filled with Greek soldiers and left outside the city gates. The Trojans fell for the trick, brought the horse inside, admired it, and retired to bed thinking the war to be over. The Greek soldiers climbed out of the horse, killed them all, and opened the city gates to their waiting comrades.

The story has been well known since ancient times, and as a result the term Trojan horse has been used for anything disguised as a gift in order for it to be taken into a normally secure location with evil intent. The horse was actually Greek of course, but the term Trojan horse is always used. The same story also gave rise to the slightly insensitive phrase "Beware Greeks bearing gifts."

The concept of a Trojan horse has proved irresistible to the modern cyber-criminal, and is a very common virus delivery method. Trojan horses are extremely easy to create, since they do not require any cleverly designed vector (delivering agent) or trigger, and people happily load them onto their own computers and run them.

To create a Trojan horse, a program is written to perform some evil, and usually criminal, act. It could be anything from deleting files, reformatting disks, or creating false data files to secretly reading confidential data and sending it back to the creator, or using the modem to dial expensive toll calls. The program is then openly posted on a web site, emailed to unsuspecting recipients, or packaged with genuine software on a CD, but with a false and inviting description. Commonly, the Trojan horse is described as a "cracked" version of some expensive and popular software package, an MP3 recording of a popular song, or compromising photographs of a popular person.

Anti-virus software *can* detect Trojan horses, *if* they are mere copies of already-caught and analyzed versions, *and* they are scanned before they are run. However, any programmer, at almost any skill level, can easily create a totally new Trojan horse. The key to avoiding Trojan horses is skepticism: do not execute software that didn't come

from a trusted source, and be very careful about whom you trust. It is very tempting to solve your budget woes by downloading free versions of software, but consider this: if a stranger walked up to you in the street and gave you a free pie, would you eat it?

Business value proposition

There is no legitimate business value to a Trojan horse.

Summary of positive issues

The software device known as a Trojan horse is known and understood, so vendors of anti-virus software can work to ensure that their software identifies and removes the offending program.

Summary of potentially negative issues

Vendors of anti-virus software need to continually modify their software to detect new and variant Trojan horses, which is a never-ending and nearly impossible task. Opening a malicious Trojan horse can lead to serious or even catastrophic consequences.

References

- M. Egan and T. Mather (2004). *The Executive Guide to Information Security: Threats, Challenges, and Solutions* (New York, Addison-Wesley).
- P. Szor (2005). *The Art of Computer Virus Research and Defense* (New York, Addison-Wesley).

Associated terminology: Virus, Spyware.

UML (Unified Modeling Language)

Foundation concept: Software development lifecycle.
Definition: UML is a standards based modeling language for specifying, visualizing, constructing, and documenting software systems.

Overview

UML is a mechanism for specifying software systems. It provides developers with a relatively intuitive methodology for the design of systems and programs. UML emerged as the leading object-oriented modeling language in the mid 1990s, and combines many popular and common concepts found in earlier methods into its modeling notations. UML provides a development structure applicable to any systems problem, and supports the system modeler from specification to implementation.

UML incorporates a large set of graphical models, including *Class* diagrams, *Use-case* diagrams, *Component* diagrams, *Deployment* diagrams, *State-chart* diagrams, *Activity* diagrams, *Sequence* diagrams, and *Collaboration* diagrams. These are used to represent different viewpoints of an overall design: the *Static* view, the *Case* view, the *State-machine* view, the *Interaction* view, and the *Model-management* view. Each view provides the modeler with a different perspective of the system. For example the static view provides details of the basic concepts from which the system is composed, whereas the activity view provides a visualization of the process flows through a model such as an activity diagram.

UML is supported by a variety of software tools both from open-source providers and from commercial organizations. These tools range from assisting in the drawing and creation of UML diagrams to code-generation systems.

Business value proposition

UML provides developers with a standardized approach to the development of system designs. The approach incorporates a wide variety of modeling techniques and diagramming methods, enabling a wide variety of systems and system types to be developed. The UML approach has been adopted as the standard development methodology by many large vendors and as a consequence there is a wide availability of consulting, software, and toolsets available. UML aids communication within project teams and the management of large projects. The wide acceptance and use of UML has increased access to human capital.

Summary of positive issues

UML is a widely used, standardized methodology for systems development. The methodology is supported by a wide array of software tools and user groups. UML is amenable to the development of a wide array of system types and environments.

Summary of potentially negative issues

UML can be conceptually difficult for programmers and developers not familiar with the object-oriented style of systems development. UML is not a formal mathematical model of development.

References

- G. Booch, J. Rumbaugh, and I. Jacobson (2005). *Unified Modeling Language User Guide* (New York, Addison-Wesley).
- M. Fowler and K. Scott (1999). *UML Distilled: A Brief Guide to the Standard Object Modeling Language* (New York, Addison-Wesley).
- J. Rumbaugh, I. Jacobson, and G. Booch (1999). *The Unified Modeling Language Reference Manual* (New York, Addison-Wesley).

Associated terminology: Object-oriented, Formal methods.

Unix

Foundation concept: Operating system.
Definition: A popular operating system for computers large and small.

Overview

It used to be that every computer manufacturer also produced their own operating system. They were totally tuned to one particular kind of computer, and gave carefully tailored control over all aspects of its operations. As computers became cheaper, their purchase price could no longer cover the high costs of such complex software development, and now the practice is very rare indeed. It seems as though every computer there is runs either Unix or Windows, and things really are almost exactly what they seem. There are a few exceptions, but only a few. Even the Macintosh operating system is now a graphical shell running on top of a standard Unix.

Unix started its life in the early 1970s as an AT&T Bell Labs project led by Kenneth Thompson and Dennis Ritchie. It was originally a very simple and basic operating system, providing only the absolutely essential features, relying on third-party applications to perform most of its tasks. It has since grown considerably, but still follows the same principle of design: a *Kernel* covering the essentials, a *Command-line interpreter*, or *Shell*, with very little intrinsic functionality, and a large collection of independent applications and components to do the work.

For many years, Unix was a strictly proprietary piece of software, with use permitted only under license from AT&T. The source code was kept secret, making many aspects of system design very difficult, and rendering the system unsuitable as a basis for academic study. Over the years, several complete "clean-room" redevelopments were undertaken, so that there now exist Unix-like operating systems that were created by people who had never seen the AT&T code, and such systems are not derived from any earlier systems. The result is free and open implementations not requiring any licensing. The most popular free versions of Unix are *FreeBSD* (from Berkeley Software Distribution), and *Linux* (based on Andrew Tanenbaum's lightweight system *Minix*, but independently redeveloped by Linus Torvalds). Sun Microsystems' *Solaris* and *SunOS* are also versions of Unix, as now is *MacOS*. It is unclear whether these new systems can really be called Unix in a legal sense, since they are distinct products, but in every other sense they certainly are.

The command-line interfaces to Unix (the shells) are characterized by exceptionally arcane syntax and a complete lack of any overall design. For example, to see the contents of a text file, the command is "cat"; to see the file with all non-text characters rendered visible, it is "cat -e." The basic command for file backup is "tar"; to obtain a listing of the files that were backed up in a particular archive one has to remember the incantation "tar xvf." The terse syntax can be very dangerous: the command to delete all files whose names begin with the four letters "temp," a common and reasonable task, is "rm temp∗," but, if it is accidentally typed as "rm temp∗" (just an extra space), every file is deleted, regardless of its name, without warning. Even experienced users spend a lot of time referring to reference material in an annoyed manner. For non-technical users, one of the many *Graphical user interfaces* (GUIs), or *Windowing environments*, is almost essential.

The fact that there is no one source responsible for the overall design of Unix-like systems, and essential utilities are created by arbitrary third parties, means that there is no uniformity to the Unix experience. Different applications will behave in different ways, with different paradigms of interaction, and different command conventions. For a long time, reliability was a major issue: if software had been contributed for free by random programmers, what level of reliability could be expected? Nowadays, however, some of the varieties of Unix are amongst the most reliable operating systems available; the commercial

pressure of millions of users surrounded by thousands of hackers has resulted in great improvements.

In order to bring together all the different versions of Unix, and at least provide some common core of functionality, so that cross-version software could be developed with reasonable ease, the IEEE developed the POSIX standard (Portable Operating System Interface for uniX, now ISO 9945). The "I" in Posix represents not the user interface, but the programmer's interface. A "Posix-compliant" system will provide a standard set of library functions with uniform functionality for software developers to make use of.

Business value proposition

The choice between Unix and another operating system depends in part upon the skill levels of the user base, the applications to be run on the system, and the technology to be used in conjunction with the operating system.

Unfortunately, Unix remains a dangerous operating system in the wrong hands. Its users are expected to know what they are doing, and it is very unforgiving when they don't. For a computer owner with some reasonable technical knowledge, or an organization with a properly trained technical staff, Unix is often the operating system of choice. For those who merely need to run visual applications (word processors, spread sheets, web and email browsers), a well-designed and controlled windowing environment hiding the arcane Unix shells may be much safer.

Many technical systems administrators do in fact like Unix for the very reason that typed command interfaces are available for manipulating specific aspects of the system, rather than having to wade through layers of pull-down menus and screens. The open-source nature of many versions of Unix also appeals to certain user groups, allowing them to construct customized variants of the operating system. Many organizations and even more individuals have adopted free versions of Unix and reduced their systems costs.

Summary of positive issues

Unix is a well-known operating system that can be used in line-command mode or, if desired, through a GUI. Free versions of Unix are widely available.

Summary of potentially negative issues

Unix is extremely hard for untrained users to understand and operate. It has very powerful instructions that can perform significant systems operations (e.g., remove all data from a disk) and is unforgiving of mistakes (there is rarely an undo option). The quality of third-party components can not always be guaranteed.

There is a highly controversial dispute between the owners of the original AT&T Unix code and the developers of other versions of Unix as to the ownership of source code and claims of infringements of intellectual property rights, although varieties of Unix produced by "clean-room" development and containing no AT&T code are common.

References

- G. Glass (1993). *Unix for Programmers and Users* (Englewood Cliffs, NJ, Prentice-Hall).
- http://www.freebsd.org.
- http://www.linux.org.

Associated terminology: Open-source software.

URL (Uniform resource locator)

Foundation concept: World Wide Web.
Definition: A uniform and well-defined form for addressing internet resources.

Overview

Anything that is available over the internet, whether it be a web page, a graphical

image, an e-commerce form, a pirated movie, or a virus, is generically termed a *Resource*. In order to access any such resource, information on how to access it (which server, which directory on that server, which file in that directory, and which protocol to use) must be provided. In order to package that necessary information in one consistent form that may be understood by all network applications, the *Uniform resource locator* (URL) was invented (by Tim Berners-Lee, then a contractor for CERN, in 1989).

URLs are the familiar "web-site addresses" that users enter into web browsers, and that are embedded in hyperlinks in hypertext documents. A URL consists of four main parts, and has this general form:

protocol://hostname/resourcelocation?parameters

All but the host-name are optional. The protocol part specifies the communications language to be used with the server to access the resource; it is most commonly "http," which is the language of web servers. The host-name indicates the internet address of the server to be used; it is typically of the form "www.company.com," but may be a numeric IP address. The resource location specifies exactly which of the many resources on that server is to be accessed, and generally has the form of a file name; if it is left out, then that server's default "home page" is usually requested. The parameters are occasionally used to provide extra information specific to an individual request.

URLs give instructions on how to find a particular resource by saying exactly where to look for it. This is a very practical scheme, but is sometimes problematical. Often web sites are rearranged, and resources change locations. Using an old URL could easily result in the wrong resource being accessed. Even more frequently, old URLs point to locations that no longer contain anything. URLs do not include expiration dates or any other means for verifying that the correct resource has been found.

A URN, or *Uniform resource name*, provides a more abstract way of identifying a resource. URNs give names to resources, and those names are supposedly guaranteed to always uniquely identify the same resource regardless of how or where it is stored. For example, a URN might be something like "urn:isbn:0-201-63346–9"; this indicates the book that has ISBN 0-201-63346–9 (ISBNs, *International standard book numbers*, are an already-established unique identifier for all books). A URN is in a sense the exact opposite of a URL – it uniquely identifies a resource, but gives no clue as to how to find it. Sadly, URNs are at present not a very practical proposition, since there is no established way to allocate or resolve them.

The term URI, *Uniform resource indicator*, is a general term covering all URLs and all URNs. If anything requests a URI, in practical terms it is really asking for a URL.

Originally, the "U" in URL, URI, and URN was intended to stand for "universal" rather than "uniform"; the change has no particular significance.

Business value proposition

URLs are a mechanism for accessing a resource on the internet and provide an organization's customers with a means of locating them. Organizations need to be careful about changing the URLs on their websites, since customers who have "bookmarked" part of a site for future reference have really just automatically recorded the URL, and may be disappointed and frustrated to receive a "404 Error" instead of the page with "Discounted Sports Equipment."

Businesses should also take care to ensure that all their URLs are carefully managed and that old "legacy" URLs from the early days of their web presence are maintained, redirected, or properly removed from servers no longer considered operational for public viewing. These legacy sites can be found by using a search engine and scanning for all public access materials.

Summary of positive issues

URLs provide a universal access mechanism for locating resources on the World Wide Web, and are an inevitable part of any web presence.

Summary of potentially negative issues

URNs (as opposed to URLs) are not currently a practical proposition. The maintenance of web sites can be more complex than expected, since changes that are not carefully managed result in customers' bookmarks becoming invalid; this creates a strong perception of unreliability and lack of devotion to customer service.

References

- P. Gralla (2004). *How the Internet Works* (Indianapolis, IN, Que).
- D. Gourley and B. Totty (2002). *HTTP, the Definitive Guide* (Sebastopol, CA, O'Reilly Press).
- http://www.w3c.org.

Associated terminology: Internet, Host, Hypertext.

Value added network (VAN)

Foundation concept: Network.

Definition: A communication network provided by a third-party vendor that also provides guarantees of service quality, security, and reliable data transmission.

Overview

Value added networks (VANs) are provided by third parties such as the common carriers (telecommunications providers) which lease secure telecommunication connections to subscribers. VANs provide a variety of services together with the connection, including support of a variety of EDI standards (such as ANSI X.12, EDIFACT, and XML), data security standards (for data encryption, a *Secure sockets layer*, etc.), multiple levels of password authorization, and communication protocols such as X.25 and TCP/IP.

Business value proposition

VANs provide secure networks capable of high-bandwidth data transmission that range from T1 at 1.544 Mbps to OC48 at 2.4 Gbps, together with service-quality guarantees and data security services. The decision to purchase may be based upon the technical need together with the financial aspects of the lease: subscribers pay a base fee plus per-service costs. The analysis of total cost of ownership has to be based upon the use of and need for the services. VANs offer a variety of services and charges are based upon their usage: the amount of data transmitted, fees for connections to other VANs, mailbox charges, help-desk service charges, data-archiving fees, service-plan fees, and user and technical training fees.

Summary of positive issues

VANs provide a secure, high-quality, professionally managed and maintained mechanism for connecting with other organizations and entities. They provide an organization with an alternative to running and maintaining its own networks.

Summary of potentially negative issues

VANs are connected over a common carrier network. While these networks can provide high bandwidth and services, internet-based network services provide a popular alternative. Internet-based networks are known as *virtual private networks* (VPNs) and can be created by corporations or through third-party VPN providers.

Reference

- L. Peterson and B. Davie (2003). *Computer Networks: A Systems Approach* (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Virtual private network, T-Carrier, Encryption, Security.

Video

Foundation concept: Digital.

Overview

Video captured by a camera compresses a three-dimensional scene into a one-dimensional stream of data. Simply looking at a scene with visible light reduces it to a two-dimensional projection; the camera scans this projection in a series of horizontal scan lines from top to bottom, and each line is reduced to a series of dots running from left to right, so the whole scene is simply a series of dots with varying colors and intensities. A moving picture is created by repeatedly scanning the same scene, making a longer series of colored dots. Converting the color and intensity of each dot into a numeric form gives a complete digital representation of one view of a moving scene.

Natural colors correspond to different wavelengths of visible light, and there is a huge range of possible wavelengths.

Fortunately, Nature took a short-cut when designing the human eye, and the color-sensing cells in the retina react to stimuli only in three broad wavelength ranges. This is why there are three primary colors: it has nothing to do with the physics of light; it just arises from the structure of the human eye. To accurately represent the color and intensity of any point of light from the human point of view, only three numbers are required, to represent the intensities in each of the three bands: red, green, and blue. Thus the storage space required for any video signal may be computed as 3 × the number of frames scanned per second × the number of horizontal rows per scan × the number of resolved dots per line × the length of the video sequence in seconds × the amount of space required to store a single number. To make a picture smooth enough to please the average viewer, each number requires only one byte, about 500 lines and 500 dots per line, and 30 scans per second are required as a minimum. This results in a total of 22 megabytes per second, or 120 gigabytes for a short movie; a very large quantity of data indeed, and an impossible bandwidth requirement for almost any network connection.

The figures of 500 rows and 500 dots are somewhat arbitrary; standard television signals are approximately 500×300 and "DVD quality" is, in the United States at least, 720×480. The resolution perceivable by the eye is much higher than that, but cameras could easily be designed with a higher resolution if needed. The figure of 30 scans per second is based soundly on biology: the eye can not react to changes quite that fast, so a higher scan rate would produce no improvement for human viewers; a scan rate as low as 20 per second would produce a very irritating flickering effect.

Twenty-two megabytes per second is an almost impossible standard to meet. Fortunately, there is a lot of redundancy in most real scenes, so compression can come to the rescue. Compression ratios of 30 : 1 may be achieved with generally acceptable results, although rapidly changing or subtly detailed scenes do suffer degradation.

The most used format for digital video is now *MPEG* (Motion Picture Experts Group), which is currently available in four versions. (The popular audio format MP3 is an abbreviation for MPEG-3.) MPEG is the standard used for domestic DVD recordings, most digital satellite television signals, and most streaming video systems. MPEG is closely related to the JPEG system for still images; it views the image as a series of 8×8-pixel blocks, which are individually compressed. When too high a compression ratio is demanded, these little squares become easily visible. MPEG also groups together small sequences of scanned images, taking the first whole (but compressed) as a *Key frame*, then recording only how subsequent frames in the group differ from the first. When the scene changes slowly, this produces great improvements in compression without noticeable loss of quality; when the scene changes quickly, it produces highly visible, strangely colored rectangular "MPEG artifacts."

This belies the deliberate myth of "digital quality." A digital signal is always an approximation to an original analog signal, and a compressed one more so. Digital video does have two advantages: it is in a conveniently computer-manipulable form, and perfect copies can be made without the "generation loss" familiar to those who have tried to copy VHS tapes. Digital video, like digital audio, has no innate lead over a corresponding analog signal in terms of quality; it is simply cheaper and more convenient for providers.

Other video formats abound. The *AVI* (Audio/Video Interleave) format introduced by Microsoft is popular in video production because it is easier to process and tends to produce better quality than MPEG, but as a consequence produces much larger

files, so it is not popular for distribution. Apple's *Quicktime* "*MOV*" format is also popular, and frequently used for short downloadable movie clips.

Business value proposition

There has been explosive growth in the use of digital video since low-cost video technologies became available during the 1990s. The newly affordable technologies include moderate-quality cameras costing only a few tens of dollars but requiring direct connection to a computer, video cameras with their own built-in recording mechanisms, and, of course, digital versions of traditional VCRs using computer disks instead of tapes. A significant increase in affordable network bandwidth and the availability of affordable CD- and DVD-burning hardware were also major influences.

The proliferation of video devices was paralleled by the development of technologies and software to allow the duplication, storage, and manipulation of the video data. These include software for the production of online videos capable of being downloaded through the internet or multicast through *Local area networks*, which has led to a large increase in the use of, if not in the quality of, on-demand instructional and training video.

The potential uses of video technologies are vast, and the delivery of video over the internet has become an area of commercial and academic interest. The long-awaited delivery of video-on-demand television is now possible over high-bandwidth networks. Wide availability of high-speed broadband internet connections is needed both to create a positive economic demand model and to facilitate the delivery of video signals with sufficient image resolution. Computer systems may be connected to large projection systems to provide a home or corporate in-house theater effect. There is great potential for future expansion in this direction; the delivery of radio over the internet and short video clips sent to cellular telephones are certainly just the first steps.

Summary of positive issues

There are several well-known and established standards associated with video. Video technologies and internet technologies are continually improving, allowing higher-resolution and larger video files to be transferred. Digital video delivered over a network is a cost-effective mechanism for the delivery of short or low-bandwidth transmissions, which is particularly valuable for training and educational purposes.

Summary of potentially negative issues

Digital video requires a large amount of storage capacity and bandwidth. Video compression when taken too far significantly reduces the quality of images.

Reference

- D. Austerberry (2002). *Technology of Video and Audio Streaming* (Woburn, MA, Focal Press).

Associated terminology: Compression, Audio, Digital, Bandwidth.

Virtual machine

Definition: A software application that perfectly imitates a real computer.

Overview

It is possible to design a program that provides a perfect imitation of the hardware of a real computer. By duplicating, as a software simulation, all the operations of a given piece of hardware, a purely software application can arrange to produce exactly the same effect. Naturally, a software simulation will be slower than the real thing, and generally less efficient in other ways, but in all important respects a piece of digital hardware that is fully understood can be

perfectly rendered in software. The result is called a *Virtual machine*.

Virtual machines provide four valuable services, relating to experimentation, hardware–software compatibility, system security, and software construction and portability.

Experimentation: when the purchase of a new kind of computer system is being considered, buyers can get a strong idea of what the new system is like to use without having to buy or borrow a demonstration model if they can run a software simulation of it on their existing computer system. When a new computer system is first designed, enormous savings can be realized by first constructing a virtual machine to test the design and iron out the major bugs before constructing any new hardware.

Hardware–software compatibility: most software runs exclusively on one particular hardware platform: PC software does not run on Macintosh hardware. Although some software manufacturers do produce versions for different platforms, this is by no means universal. A virtual machine allows one computer to behave like another, and therefore run software designed for that other system. For example, a Macintosh computer can run a virtual machine that imitates a PC, and that virtual PC can then run any PC software.

System security: virtual machines are not limited to imitating *another* kind of computer. It is quite possible, and common, for a PC with Windows to run a virtual machine that imitates a PC with Windows. The reason for this is that untrusted software (perhaps it is suspected of being a Trojan horse or bearing a virus, or perhaps it just doesn't work very well and keeps crashing the computer) can be run on the virtual machine; it will behave exactly as it would on the real computer, but will be incapable of doing any harm, because its environment is simulated. It can't erase the real disks or crash the real computer; it can only harm the simulation, and the simulation can be restarted in a fraction of a second.

Software construction and portability: this is currently the most popular use of virtual machines. Sometimes, the basic way that modern computers work makes certain software solutions very difficult to implement, and designers conclude that it would be much easier if they could design a whole new kind of computer just for this one program. Virtual machines make this into a viable solution. The new computer, idealized for the solution of this one problem, and perhaps totally infeasible for real construction, is designed and implemented as a software simulation. The difficult problem is then solved with a program written for that new computer, and the computer simulation packaged together with that program is the product. This is an especially valuable technique when new programming languages are designed and first implemented. Once a product has been developed by this method, it is much easier to produce versions for other hardware platforms: only the virtual machine needs to be *Ported* to another kind of computer; the program that it supports, the major part of the product, remains unchanged. This is part of the philosophy of Java. All Java programs run on a virtual machine known as JVM, the Java Virtual Machine. All Java programs are automatically cross-platform because an equivalent JVM has already been written for all major hardware platforms.

Business value proposition

Virtual machines are software programs that simulate the hardware of a real or theoretical computer. The best known use of virtual machines is in the Java programming area, where Java programs run on a Java Virtual Machine (JVM) and as such are highly portable, since the JVM can be re-written for any hardware platform. This prevents the problem associated with many

programming languages, in that a specific version of the language will only run on a single machine and locating or writing a new compiler for the language so that it can be used with a new computer is difficult, expensive, and time-consuming. The virtual machine concept is one of the reasons behind the popularity of Java and its development environment.

A second popular use of the virtual machine concept is to allow one machine to simulate another machine, such as a PC being simulated on a Macintosh, and consequently PC applications can run on the Macintosh through the simulation.

Another use that is gaining popularity amongst network managers and chief security officers is the ability to use virtual machines as a further layer of defense against intruders and malicious software attacks. A virtual machine can be made to simulate the machine upon which it is in fact running and therefore, should the software turn out to be malicious, it would only be attacking the simulation and not be able to do any real harm.

Virtual machines may be used in organizations as mechanisms to determine how a software package would work upon another hardware platform. For example, an ERP system module may represent a large investment and a company may wish to assess whether the system would perform as desired upon a new platform prior to its purchase. This would allow performance issues to be considered and aid the organization in performing capacity planning.

Summary of positive issues

The theoretical aspects of virtual machine development are well known amongst computer scientists. Virtual machines exist to support a variety of languages and of platforms that aid the portability of applications. Virtual machines can be used as an extra security layer.

Summary of potentially negative issues

Training as a computer scientist is required if one is to create a useful virtual machine. The creation of a virtual machine requires the expenditure of focused specialized resources and may take some time to complete for a challenging simulation problem. Developing a virtual machine may be a lengthy exercise and should not generally be considered a quick fix for a critical problem. Virtual machines are less efficient than the real hardware system that they simulate.

Reference

- J. Smith and R. Fair (2005). *Virtual Machines: Versatile Platforms for Systems and Processes*, Morgan Kaufmann Series in Computer Architecture and Design (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Java.

Virtual memory

Foundation concepts: Memory, Disk.
Definition: A system that makes a computer appear to have more memory available than it really has.

Overview

A *Virtual memory* system usually has two components: *Memory mapping*, which allows many applications to run concurrently, with all of them able to make use of all of the computer's memory without conflict; and *Paging*, which allows applications to make use of more memory than is actually installed in the computer. Virtual memory is an essential component of any general-purpose operating system.

Memory mapping is a service provided by special-purpose hardware, which requires special support from the operating system software. A computer's memory is viewed as a collection of literally millions of individual *Locations*, each capable of storing a single simple item of data such as a small number or an alphanumeric

character. Each of these locations has a fixed numeric address, and all accesses to memory must specify the numeric address of the location to be accessed. For example, a database application may store the current count of the number of records in the database in location number 301 440, and might store the list of employees' social-security numbers in the region of memory between locations 1 241 780 and 5 241 779; a word-processing application might store the name of the file it is working on between locations 301 300 and 301 526, and the contents of that document starting at location 2 000 000. If those two applications are ever to be run concurrently, their memory usages would conflict.

Memory mapping hardware performs on-the-fly location translations for each running application. Every time a memory location is accessed, the memory mapping hardware looks up the location to be accessed in a mapping table, finds the alternate location that that access should be translated to, and causes that address to be used instead. The operating system sets up a mapping table for each application as it is started, and ensures that all applications' memory accesses will be translated to unique locations, avoiding all conflicts. A typical modern processor may make 100 000 000 or more memory accesses per second, so the careful design of memory mapping hardware to work at these speeds is critical to system performance. Without memory mapping, it would be not quite impossible but very difficult and inefficient to have more than one application running concurrently through *Time-sharing*.

Paging is a service provided by operating system software, which requires special support from the memory mapping hardware. Paging allows a computer system to act as though it has much more memory available than is actually installed. It does this by selecting certain things that are currently stored in immediate-access memory to be *Paged out*, and stored on disk instead, thus freeing up that area of memory to be used for something else. Of course, when an application needs to access data that has been paged out, it must be slightly delayed while that data is *Paged in*, and brought back into memory from disk. Since disk drives work approximately 1 000 000 times more slowly than normal memory, the system must take great care in deciding which areas of memory can reasonably be paged out.

Fortunately, most applications on a personal computer are almost completely *User-bound*. This means that they spend the vast majority of their time just waiting for a human user to press a key or click the mouse. Modern computers are so fast that a whole application can sometimes be paged out, or suspended to disk, while waiting for user input, and restored so quickly when that input is provided that the user is totally unaware of any delay. With a well-designed paging algorithm, a modern operating system can easily over-subscribe its use of memory by a factor of ten or more, allowing a computer with 128 MB of memory to behave like a computer with 1280 MB of memory.

Paging is so called because memory is divided into a number of small regions, usually of 512 or 4096 bytes each and called *Pages*. For efficiency, it is not individual memory locations that are paged out, but whole pages at once. Paging is not essential for any operating system, and, when computationally intensive applications, or *Real-time* applications (those that must be able to respond to emergent conditions immediately), are running, it may be desirable to turn paging off, either for individual applications or for the whole system. Not all operating systems provide fine control of paging, and that is an important aspect to consider when choosing a system.

Business value proposition

The technical aspects of virtual memory are fixed by the hardware and operating system manufacturers and thus there is little that the end user can do in this regard. Usually only the most minor of adjustments are possible. If a system does not offer virtual memory capability, that may be a serious impediment to its candidacy. For some applications, such as small embedded systems, virtual memory is not appropriate, but that assessment should be made only by technical experts.

Summary of positive issues

Virtual memory provides systems with the ability to service multiple applications and users concurrently. Memory mapping allows more flexible use of memory and as a consequence optimizes memory resources. Paging allows computers to extend the capability of their memory by bringing in and out of the page file the resources required by applications as the operating system switches between them.

Summary of potentially negative issues

Memory management is a highly technical aspect of computer engineering and should not be interfered with by non-specialists. Paging is also a complex aspect of the design of a memory management system, so any adjustments to the paging mechanism such as turning it off need careful consideration and technical advice is recommended.

References

- D. Groth (2003). *A+ Complete* (Alameda, CA, Sybex Press).
- M. Gorman (2004). *Understanding the Linux Virtual Memory Manager* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Operating system.

Virtual organization

Foundation concepts: e-Commerce, e-Business.
Definition: A virtual organization is an organizational structure under which value-chain activities may be located anywhere, either owned by the organization or sourced from a third-party provider, and connected together through information technology.

Overview

The concept of a virtual organization arises from the notion that all aspects of a traditional company's value chain can be either outsourced or provided by a third party and managed through information technology. The goal of creating a virtual organization is to enable flexible production to occur, resulting in what Davidow and Malone have termed the *Virtual product*.

The term *Virtual* in both of these contexts is derived from the use of the word in computer engineering (virtual memory, virtual reality, virtual terminal, etc.), where it denotes some form of nonexistence or abstraction from reality. A virtual organization is one in which some or all of the core processes have been outsourced to a third-party service provider. A simple example would be a company that receives orders for some product from the public through the internet, buying the product directly from their suppliers only when full payment has been received, using a delivery service such as FedEx or UPS to send the product on to the customer. Even the processing of payments may be outsourced, allowing the company to provide a vast range of products by coordinating and managing processes while performing hardly any processes themselves.

Similarly, a company selling virtual products may advertise a wide range of highly customizable products, such as computer systems with almost infinitely variable options selected by the customer. Such a company may have absolutely no stock in hand, simply ordering the required parts

once an order has been received and full payment made, then assembling them and shipping the product out. This strategy may provide great savings in inventory and warehousing costs.

In practice, pure virtual organizations exist only in the service sector (e.g., consulting), while in the manufacturing sector a close approximation can be found in organizations that utilize techniques associated with mass customization, as advocated by researchers such as Joseph Pine. The term *Agile* has also been associated with discussions surrounding the concept of virtual organization. Steven Goldman and other pioneers in this area have promoted the agile organizational model, in which products and production resources are designed to allow continuous product evolution as market demands change.

Business value proposition

The concept of a virtual organization is in operation to some extent in the majority of organizations that use a third-party service provider for any activity and manage that relationship through technology. The *pure* virtual organization concept in which every function is outsourced has not been embraced by many organizations, but as the ability of an organization to monitor its outsourced functions through sophisticated systems increases the adoption of this model will increase. The concept of virtual organization allows organizations to concentrate on deriving the best value from each service provider, reducing the operating expenses through technology, and using the managerial resources to execute the business plan effectively.

Summary of positive issues

The concept of a virtual organization allows greater flexibility and partnership with "world-class" service providers. The application of leading technologies allows greater agility in the company, rapidly adjusting the production of manufacturing or provision of service to align with demand or market fluctuations.

Summary of potentially negative issues

The concept of virtual organization incurs a high overhead in terms of command and control, information technology, and relationship management. Partners may find it difficult to provide services to the virtual organization with the flexibility desired. Virtual organizations are dependent upon their partners to perform core and non-core functions successfully.

References

- W. Davidow and M. Malone (1992). *The Virtual Corporation* (New York, Harper Business).
- J. Pine (1993). *Mass Customization: The New Frontier in Business Competition* (Boston, MA, Harvard Business School Press).
- S. Goldman, R. Nagel, and K. Preiss (1995). *Agile Competitors and Virtual Organizations: Strategies for Enriching the Customer* (New York, Van Nostrand Reinhold).

Associated terminology: Business process re-engineering.

Virtual private network (VPN)

Foundation concept: Network.

Definition: A network that has no independent physical existence, but exists only as a set of connections over a larger real network, usually the internet.

Overview

If an organization has offices in London, Sydney, Grytviken, and New York, the construction of a private network connecting the four would be prohibitively expensive. A worldwide network, the internet, already exists and provides public connections between any two points. A *Virtual private network* (VPN) uses the existing infrastructure of the internet, or any other

physical network, as the foundation of a new free network. A VPN could almost be viewed as a parasite on a real network; it exists only as an abstraction, an alternate view of parts of the real network.

In a *Client–server*-based VPN, a server is set up at some designated location to represent the private network. All other *Nodes* (sites on the VPN) run special-purpose client software, which makes a standard internet connection to the server, and maintains that connection for as long as the VPN is active. All messages that are supposed to be transmitted on the private network are actually transmitted to the server, and the server retransmits them to the appropriate destination site. This provides the same functionality as a real private network, but with slightly less efficiency (all messages must make two trips instead of one), and virtually no cost.

In a *Peer-to-peer*-based VPN, all sites on the private network make and maintain direct internet connections to some or all of the other sites. Messages that are intended to be sent on the private network are instead sent to the same destination over the existing network. When a VPN has more than a small number of nodes, it is not practical for each to maintain a connection to all of the others. In such cases, a connectionless protocol such as UDP is more likely to be used than TCP; the end results are the same with suitable software support.

There are two advantages that a true private network would have. One is security: if an organization is completely in control of its network hardware, it can physically shut out spies and unwelcome communications from external sources. The same level of security can easily be provided with the proper use of strong encryption and firewalls to exclude unauthenticated transmissions.

The second advantage is control over reliability. The organization owning the network hardware can decide exactly how important reliability is, and allocate whatever funding they see fit to maintenance and redundancy. When the network is virtual, and exists only over public internet connections, no such control is possible. The main connections of the internet, called its *Backbone*, are extremely reliable, but the end connections provided by local telecommunications companies and ISPs can be problematical.

Business value proposition

VPNs provide an alternative mechanism for organizations that wish to take a higher level of control of their network, extending and managing their network beyond their corporate boundaries. VPN technologies have evolved considerably since the 1960s and 1970s when the options were to use a dial-up modem or lease a line from the telephone company. The "leased-line problem," however, remains the same today as it was in 1960, since the costs associated with this approach are still the biggest barrier to their use, and the combinatoric explosion of "peer-to-peer" connections makes the scalability of this solution unsustainable for all but the largest of organizations.

The business case for a VPN is based upon security, speed requirements, and the quality of service provided by the network. The use of an internet-based solution with off-the-shelf hardware and software solutions can create a highly effective solution applicable and acceptable to the majority of organizations wishing to use a VPN. These technologies include firewalls (hardware- or software-based), the *Secure sockets layer* (SSL), and other encryption protocols. The use of widely available technologies allows a VPN to scale well, while maintaining a technological basis that is generally compatible with the remainder of the corporate network architectures.

Summary of positive issues

Leased-line VPNs provide secure, reliable networks that allow the quality of service

to be controlled by the membership of the network. Internet-based VPNs provide lower-cost solutions for companies that can work within the parameters of the internet (speed variances based upon the network's bandwidth and levels of use by others, etc.). VPNs can be built out of well-known, well-understood, and fully documented protocols and technologies.

Summary of potentially negative issues

Leased-line VPNs can be expensive, depending upon the speed and other requirements of the members. Public internet-based VPNs are less secure than leased-line systems and issues such as speed might not be guaranteed by the ISP. The creation of a VPN requires a significant amount of technology training and an understanding of networks.

Reference

- R. Yuan and W. Strayer (2001). *Virtual Private Networks: Technologies and Solutions* (New York, Addison-Wesley).

Associated terminology: Internet, Bandwidth, Security, Internet protocol, TCP.

Virtual reality

Definition: Virtual reality is a computer-generated immersive three-dimensional simulation of an environment.

Overview

Virtual reality has its roots in the work of Morton Heilig, who in 1957 invented the *Sensorama Machine* (US patent 3 050 870 granted in 1962) that attempted to create a three-dimensional (3D) cinematic experience including smell, stereo sound, and even seats that vibrated. In 1960 he invented the first head-mounted display (HMD), known as the *Telesphere Mask* (US patent 2 955 156 granted in 1960), which provided 3D vision combined with stereo sound. During the subsequent 40 years, researchers such as Ivan Sutherland developed the fundamental theory of computer graphics and 3D imaging which underlie virtual reality. During the 1960s and 1970s virtual reality received a great deal of attention in military circles as a means for creating aircraft and battlefield simulators and other military training environments. During the 1990s the advent of more powerful and flexible computing equipment allowed the practical implementation of creative projects such as *CAVE*, a room-sized high-resolution 3D video and audio environment developed at the University of Illinois at Chicago in 1991, and *Jack*, a 3D interactive environment for controlling detailed models of human figures, which was developed at the University of Pennsylvania.

Business value proposition

Since the 1950s the use of virtual reality has moved from the research laboratory into the workplace, factory, and home. The development of products such as automobiles and airplanes would be very difficult without the use of such technologies. The technology is used as a modeling aid in computer-aided styling, factory design whereby models such as Jack are used to examine work practices, including the physical stresses placed upon workers, and sophisticated simulators are used for the training of pilots. Virtual reality allows manufacturers to examine their product ideas more thoroughly, more quickly, and in more depth than is possible with other forms of modeling.

Summary of positive issues

Virtual reality technology is commercially available and software provides platforms for a variety of industries and processes, including medical systems, space applications, manufacturing, aviation, and military applications.

Summary of potentially negative issues

Virtual reality can be resource-intensive and may require specialized hardware for high-end performance applications. Distributed virtual reality has the added complication of potentially requiring a high-bandwidth network between the entities interacting in the simulation.

Reference

- W. Sherman and A. Craig (2003). *Understanding Virtual Reality: Interface, Application, and Design*, Morgan Kaufmann Series in Computer Graphics (San Francisco, CA, Morgan Kaufmann).

Associated terminology: Artificial intelligence, Video.

Virus

Foundation concept: Security.

Definition: A virus is an illicit piece of software, deliberately installed on a computer by an unauthorized user without the owner's consent. The usual purpose of a virus is to cause damage to, or destruction of, files.

Overview

Computer viruses are now universally known, but generally very poorly understood. The original viruses were self-replicating programs. Once a virus had been delivered to a computer, it would make copies of itself to thwart attempts at removal, and then send copies of itself to other computers (using email contact lists, shared document services, and many other means). The name "virus" is used because of this strong resemblance to the behavior of a biological virus. The term is now used much more loosely, generally meaning any software deliberately run without authorization.

A successful virus could be quietly sent to just a few vulnerable computers, and then spread itself throughout the world with virtually no chance of its origin ever being traced. In 1988, a virus known simply as "The Internet Worm" virtually shut down the entire internet within a few hours of its release.

Originally the term "virus" was reserved specifically for illicit software that became attached to some already-existing application or executable code, which would become active only when that infected piece of software was run. Conversely, a "worm" is self-contained, and does not need to infect anything else. Worms are typically delivered and immediately activated through some security flaw in the operating system. When activated, they immediately attempt to retransmit themselves by the same means to other computers in the vicinity. Recently the distinction between "virus" and "worm" has become blurred, and "virus" is generally taken to cover all such things.

Delivery: any computer with any kind of internet connection is likely to be vulnerable to viruses. Even if the computer is used only for browsing the web, it will still have many hidden internet connections. An instant of carelessness by the programmers who created the operating system can leave a security hole that will eventually be discovered and exploited, allowing viruses to arrive through "back doors" that the computer owner didn't know existed. Viruses may also arrive hidden inside web pages that are being browsed, or in email attachments. Any time two computers are connected, even indirectly, there is a potential for a virus to spread from one to the other. Any time files are shared (using floppy disks, CDs, or DVDs, for example), those files could be delivering a virus.

Activation: contrary to popular belief, simply having a virus on a computer does absolutely no harm at all. A virus is a program, a piece of software; it can not do anything unless it is executed. With older, poorly designed computer systems, programs could easily be run automatically without the owner's consent. If an email

or a web page contained a piece of software, just opening the email or browsing the page would be enough to start that software running. With more recent operating systems, user-selectable safety settings exist. The user can decide that no software ever runs automatically, or perhaps only software from trusted sources. The user must be aware of these settings, and deliberately activate them, but they do provide some amount of protection.

Unfortunately, a virus delivered through a "back door" can still often be activated automatically. There is a sadly common software fault called a "buffer over-run." In simple terms, a software application with an internet connection has to set aside some memory for incoming messages to be assembled in. If the incoming message is longer than it should be, and the programmer left out all the checks, an incoming message can over-run the boundaries of the assembly area and become part of the application itself. The checks to prevent buffer over-runs are exceptionally simple, and the problem of buffer over-runs is well known. For a commercial programmer to allow a buffer over-run is an act of carelessness bordering on gross incompetence, but, sadly, buffer over-runs are still probably the most common means of entry for a virus.

Purpose: the general expectation is that a virus is intended to erase or modify essential data, or even destroy computers, to cause great mischief and financial loss. While this is definitely true in some cases, the most common virus is simply an "ego trip." Somebody wants to cause just enough annoyance to be noticed, and earn them bragging rights amongst the community of "hackers," as they like to call themselves. The fact that great harm is not usually intended should be of little comfort. Virus software is often very poorly designed, and can cause a lot of unintended damage. Most importantly of all, a virus *can* be truly malicious. Once an illicit program is running on your computer, there are very few limits to its scope. It *could* erase every byte of data in the whole system; it *could* transmit all of your personal data over an internet connection to anywhere in the world; it *could* make your modem dial a very expensive 1--900 number; or it *could* just pop up a message saying "Got you!"

Quite frequently, when a virus gets into a computer, it will secretly take over that computer, allowing the virus creator to send commands and mount other attacks, or store illegal copies of copyrighted material on a computer without the owner ever knowing. Most of the seriously damaging attacks are mounted indirectly from computers that have been taken over by a virus, so that the originator of the attack is virtually untraceable. This could have unpleasant legal consequences for the innocent computer owner.

It is not impossible for a virus to cause physical harm to a computer, but it is very unlikely, and extremely rare. Just as with a real biological virus, if it kills its victim, it has no opportunity to spread itself to others.

Creation: virus creation can be a highly skilled and intellectually demanding job, but, as with most criminal enterprises, the miscreants are usually completely unskilled users of widely available "virus kits." Finding a security flaw in a commonly used application, and then working out a way to exploit it, is a very difficult task. Unfortunately, it rarely has to be done. Responsible software companies are always testing their own products, and, when they find a flaw, the only responsible action is to release a "patch" (a small program that automatically repairs the flaw) to their customers. Once such a patch has been released, it is much easier for a virus designer to work out where the flaw is, then release a tailor-made virus, knowing that most computer users will not install the patch in time to be safe.

By far the largest group of viruses consists of those created by people who either

blindly follow a script for virus creation, or take an existing virus and insert their own names into it. Thus all such viruses tend to do the same thing, and often don't work (either doing nothing, or being far more destructive than was intended).

Business value proposition

Even a virus that does no tangible harm has a negative impact on the bottom line. It takes time and resources to remove a virus from an infected computer, and they must always be removed, because seeming harmless is no guarantee of being harmless. A virus consumes resources, tapping the computer's processing power and memory, and uses up valuable communications bandwidth. Deliberately destructive viruses are another matter altogether and can paralyze a company through its computing resources.

The only way to be completely safe from viruses is either not to have a computer, or to have a computer but no modem, and no internet connection, and never to accept floppies, CDs, or DVDs from anyone. That is clearly not a practical approach. To be *relatively* safe from viruses, there are a few simple precautions that organizations and individuals can follow.

- Do not buy software from companies that have a history of exploitable software.
- Turn on all the user-selectable security levels.
- Disable JavaScript in web pages and other active software applications.
- Do not open an email attachment from somebody you do not know unless it has been scanned by an anti-virus software package.
- Do not access floppies or CDs that have not been scanned for viruses.
- Use a reliable *Firewall*.

No matter how good your security is, the only responsible attitude for corporations is to expect to be the victim of a disastrous attack and take appropriate steps in advance. These include encrypting all sensitive data stored on a network-accessible computer, and ensuring that all valuable data is effectively backed up: copied onto some removable media (CD, DVD, tape, etc.), and physically removed from the computer and the site. The only data that can not be destroyed by a virus is data that is not on a computer.

Most importantly, corporations and individuals should always install security patches as soon as they are released by the software company that created the operating system and other applications in use upon the organization's computers. It is also important to make sure that the patch being installed is really from the actual original software company. (Another form of attack, known as the "*Trojan horse*," is to send out free software that claims to be something useful, perhaps a security patch even, but is in fact just a virus in different packaging.)

Special-purpose "anti-virus" software can be effective, but is not the silver bullet that it is often believed (or hyped) to be. Companies that create anti-virus software monitor the viruses that are flying about the internet. Whenever they detect a new one, they work out a way to detect and disable it, and then send out a (usually free) update to all of their customers. Except for a few rare cases, anti-virus software can protect only against a virus that has already been caught and analyzed. If a company or an individual is the victim of a new virus (and most victims do fall in the first few days after a virus release), the anti-virus software will probably not be able to do anything until the "antidote" for the new virus has been released.

It should be noted that personal computing devices that are not "company issue," such as home PCs, need to be considered as part of the company's security assessment, since executives who take home sensitive

or vital corporate data may be vulnerable in their less secure home-computing environments.

Summary of positive issues

Theoretically, it is possible for there to be a good virus. A skilled programmer could use all the techniques of the virus creator to produce software that disseminates itself about the internet, "attacking" every computer it finds, by repairing all known security flaws and killing any active viruses. Such a product is unlikely to exist, since nobody would make any money from its creation, and there is a significant legal liability if the creator makes a mistake, and the "good" virus does some harm. There is also a significant legal liability if the good virus works perfectly, because it would be illegal in most jurisdictions. Practically, there are no positive issues other than to say that governments are actively creating and developing legislation to prevent, or at least discourage, the malicious use of viruses or other software technology.

Summary of potentially negative issues

The potential negative impact of viruses upon companies, individuals, and governments can not be understated and antivirus measures need to be in place prior to an attack occurring. It has been estimated that viruses and their relatives have a total annual cost (for repair, removal, and incurred losses) of over $13 billion in the United States alone, although there are no truly reliable statistics.

References

- M. Egan and T. Mather (2004). *The Executive Guide to Information Security: Threats, Challenges, and Solutions* (New York, Addison-Wesley).
- P. Szor (2005). *The Art of Computer Virus Research and Defense* (New York, Addison-Wesley).

Associated terminology: Email, Encryption, Internet, Trojan horse, Spyware, Law cross-reference.

Visual Basic

Foundation concept: Programming language.
Definition: A simplified programming language popular for low-volume data-transfer tasks and creation of Windows user interfaces.

Overview

The name Basic is an acronym for *Beginners'* All-purpose Symbolic Instruction Code, which was created at Dartmouth College in 1963. It was designed to make programming possible for those who do not have the time to learn it. Owing to the altruistic efforts of its designers to make it widely and freely available, Basic was already well known when the microcomputer revolution began. Because it is so small and simple to implement, it became not just the language of choice, but the only language available for users of many small computers. A more "serious" language would not have been able to fit on computers with such limited memory capacity. By the 1980s, Basic was probably the most widely known of all programming languages.

The most common tasks for nontechnical commercial programmers (that is, the vast majority of all programmers today) are the transfer of data between two applications or between an application and a human user. This kind of programming does not require great efficiency or the ability to express advanced algorithms; it is relatively straightforward data processing and user-interface control. Data produced by one application may need to be reformatted before it can be used by another; human users now expect to be able to provide input to applications through convenient and well-laid-out dialogs and windows, and also expect to see the results in the same way.

Any operating system that expects to gain popular acclaim is going to need to provide some way for people to be able to carry out this kind of programming easily. There are not enough technically trained software engineers in the world to meet the demands created by the ubiquity of computers, so a non-technical programming language is needed. Since one of Microsoft's earliest products was a popular version of Basic for microcomputers, it is hardly surprising that they chose Basic as the basis for this language. *Visual Basic* is a highly extended version of Basic, designed particularly to work with the Windows operating systems, performing data processing and interface tasks for non-technical programmers.

Edsger Dijkstra, one of the universally recognized fathers of computer science, said of Basic (Basic in general; Visual Basic did not exist at the time) "*It is practically impossible to teach good programming to students that have had a prior exposure to BASIC: as potential programmers they are mentally mutilated beyond hope of regeneration*." He was partly joking. When used for its intended purpose, data transfer and implementation of graphical user interfaces, Visual Basic is an immensely successful tool, and it is hard to argue with success. The only problems arise when dialects of Basic are used beyond their scope.

Business value proposition

Visual Basic provides end users and non-technical programmers with a tool for developing their own programs quickly and easily. The language does not require advanced training and can be quickly assimilated and used. The language can also be used to develop prototypes quickly; these are non-production versions of software that simulate the interfaces and basic functionality of a system that may potentially be developed.

The adoption of Visual Basic frees the IT organization's trained and expensive developers from having to create every program for every user. This is especially important because many Visual Basic programs are used to develop reports and are applications that are used once only or for a short period of time.

Summary of positive issues

Visual Basic is easy to learn and widely available. It was designed to work in a Microsoft environment, but applications can be ported over to other operating systems through special software. Visual Basic programs can be developed by end users, which frees technical programmers and other resources to be used on other projects.

Summary of potentially negative issues

Visual Basic is not suitable for large, complex, or technically sophisticated projects. Visual Basic is primarily designed to run on Microsoft operating systems and extra resources need to be deployed to run these applications in other environments.

Reference

- F. Balena (2002). *Programming Microsoft Visual Basic.NET (Core Reference)* (Redmond, CA, Wintellect).

Associated terminology: Fortran, Cobol, C++, Java.

Voice over IP (VoIP)

Foundation concepts: Audio, Digital.

Definition: Speech and other low-bandwidth audio, digitally encoded and transmitted over a network by computers substituting for normal telephone equipment.

Overview

Capturing sounds and converting them, in real time, into digital signals that could at any time be converted back into audible form and replayed is a very simple task.

For most people with an internet connection, data may be transmitted to any point in the world quickly, simply, and without any additional cost. Putting those two facts together results in *Internet telephony* or *Voice over IP*.

A computer with an internet connection, a microphone, and a loudspeaker effectively gives free long-distance and international telephone calls to any other similarly equipped computer. Because human speech remains completely intelligible under severe bandwidth limitations (3 kHz is quite good for a regular telephone call), there is no requirement for a fast internet connection; a bi-directional 3000 bytes per second or 24 000 baud is enough for normal telephone quality (dial-up modems usually provide the equivalent of 56 000 baud).

With even a standard speed local area network, the bandwidth may be increased to give speech reproduction with much higher fidelity, and still carry a great many concurrent conversations, making intra- and inter-network conference calls a simple and cheap reality. With the data being already in digital form, making a perfect recording of the entire conference is a trivial feature to add.

In recent years, "web phones" have emerged. These are small devices looking just like a traditional telephone, but containing a microprocessor and network connectivity hardware. These devices may be plugged directly into a DSL, cable modem, or other connection, and provide internet-based telephone service without the need for a computer.

VoIP, an acronym for voice over internet protocol, is a standard protocol that runs on top of IP (internet protocol, the main transport for internet traffic) to implement internet telephony. This is a rapidly growing market, and new protocols are appearing very frequently. A currently popular direction is to combine internet telephony with *Peer-to-peer* technology.

Internet telephony is not the panacea it may seem to be. Unlike the *Circuit-switched* regular telephone network, internet traffic is *Packet-switched*. This means that various parts of a single transmission may be transmitted by different routes and at different speeds, and thus might not reach the receiver at the ideal rate. When this happens, the result is either an extremely annoying lag in reception, or the sound becomes "choppy" or stuttering, which can render speech incomprehensible. Another minor point for consideration is safety: the regular telephone network has its own power source; after natural disasters or during bad weather there may be power outages, but often the telephones still work. Computers need an operating power supply, so they can not provide telephone service during power outages.

Business value proposition

The use of VoIP technologies is growing as they become more advanced, broadband reaches high degrees of penetration, and interface technologies become easier and more convenient to use.

The basic calculation of the total cost of ownership is based upon the need to have a physical connection, a computing device, and a "telephone" interface. For a large organization in which the majority of the work force uses a computer in performing everyday business tasks, the addition of VoIP carries a low overhead when only straightforward telephone connections are required. However, more complex requirements such as call waiting, voice-mail boxes, call forwarding, and teleconferencing make additional demands on network bandwidth and storage capacity. The major impediment to the adoption of VoIP for small organizations is the cost of the equipment relative to other options such as cellular telephones and land lines.

The network administrator and other corporate officers need to address several other issues that may affect their network through the use of VoIP, including the impact VoIP would have in terms of bandwidth requirements. A network with little spare capacity may experience severe problems when extra VoIP requirements are placed upon it. Quality of service is affected when an echo is present on the "circuit"; this can normally be solved by adopting acoustic or digital echo-canceling technology. Overall, it is important to measure the quality of the speech over the network and several vendors provide tools for speech performance measurement, typically using the G.711 standard as a benchmark.

Summary of positive issues

VoIP technology continues to advance in quality and ease of use. The running cost once the physical equipment and network service have been purchased is low.

Summary of potentially negative issues

VoIP works only when the host device is powered up. Sound quality may not be at an acceptable level for some users. VoIP requires capital outlay to provide a hardware and software platform for operation.

Reference

- T. Wallingford (2005). *Switching to VoIP* (Sebastopol, CA, O'Reilly Press).

Associated terminology: Compression, Internet protocol, Audio, Digital.

W3C (the World Wide Web

Foundation concepts: Internet, World Wide Web.
Definition: In their own words, "The World Wide Web Consortium (W3C) is an international consortium where member organizations, a full-time staff, and the public work together to develop web standards" (http://www.w3.org/Consortium/).

Overview

The W3C was founded in 1994 by Tim Berners-Lee, the inventor of the World Wide Web, and MIT in collaboration with CERN, DARPA, and the European Commission. The organization aims to develop inter-operable technologies and provide specifications, guidelines, software, and tools to help the web to reach its full potential. A primary aim of the consortium is to develop non-proprietary standards and promote inter-operability. The consortium is international in scope, has offices in fourteen locations around the world, and draws its membership from around the globe (http://www.w3.org/).

The W3C is behind many of the technologies that are in common global use, such as HTML, SOAP/XMLP, URLs, and XML.

Business value proposition

The W3C is an organization that provides leadership in the area of web technologies. Individuals and corporations can become members and, through working committees, help to develop the formulations of future web-based technologies.

Summary of positive issues

W3C provides a forum through which web-based technologies are developed. Organizations are able to join the consortium and work with the committees to innovate and create new technologies.

References

- http://www.w3.org/.
- MIT, 32 Vassar Street, Room 32-G515, Cambridge, MA 02139, USA.
- ERCIM, 2004 Rue des Lucioles, BP 93, 06902 Sophia-Antipolis Cedex, France.

Associated terminology: Hypertext, URL, XML.

Walk-through review

Foundation concept: Software development lifecycle.
Definition: A walk-through review is a peer-based review of a software development process or product.

Overview

Walk-through reviews are performed on a software component or on an aspect of the software development process by a "peer" group. Group members can not be from the team that developed the product or process and are removed from the evaluation process. The aim of the review is to examine the process or product from multiple perspectives. An example review team might be composed of engineers representing different technical disciplines providing different perspectives: a software designer, a software interface designer, and a hardware engineer. Alternatively, a review team may be formed by end users, customers, and systems analysts, performing a review along business performance lines. The goal of walk-throughs is to assess the product or process and, through a moderator, provide critical positive feedback rather than criticize the process manager, developer, or software designer.

The review process is meant to be informal and not confrontational in nature. While there are many different ways of performing the review process, the approach typically involves the developer of the software presenting the specification and the system to the peer-review team. The team then acts as *surrogate customers* and traces the system requirements through the development to the system that customers or

end users would receive. The team also aims to be objective in determining the *usability aspects* of the system, including the human factors, the interface experience, and other aspects of the software as presented to the user. The third aspect of the team's role is to ensure that the developers followed the methodology and standards as required by the vendor and the customer. Having documented the outcomes of these review categories, the results are presented back to the development team together with an evaluation of whether the system needs to undergo another walk-through review before being released for use.

Business value proposition

The objective of walk-through reviews is to provide intellectual input to a process or product quickly and effectively. Walk-through review teams can be drawn from a variety of skill bases and the mix changed according to the needs of the problem. The team may be put together rapidly, the assessment process executed in a timely manner, and the feedback quickly presented to the process owners, completing the review cycle.

Summary of positive issues

Walk-through reviews provide a rapid validation and verification tool for managers and organizations. The teams performing the review can be put together on an "as-needed" basis. Review cycle times can be very brief and reviews may be performed by different teams for different aspects of the lifecycle.

Summary of potentially negative issues

If reviews are too informal the results may be inconsistent. The team structure and skill sets might not be appropriate for the task, and thus managerial care is required in team selection. The review process should be semi-formal and requires a moderator who understands the technical issues and buffers the developers and their egos from direct criticism.

References

- E. Yourdon (1989). *Structured Walkthroughs* (New York, Yourdon Press).
- M. Deutsch and R. Willis (1988). *Software Quality Engineering: A Total Technical and Management Approach*, Prentice-Hall Series in Software Engineering (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Software development.

Waterfall model

Foundation concept: Software development lifecycle.
Definition: The waterfall model is a software development lifecycle model.

Overview

The *Waterfall model* developed by Royce in 1970 was the first software development model to address all phases of the software lifecycle, capturing each phase before "cascading" its results into the next. The model is usually drawn as a series of descending steps, drawn as boxes, with the results of a higher box flowing into the next lower one, suggesting its name "the waterfall model."

While there is no "official" number of phases in a waterfall model, a basic version begins with a *Requirements definition* phase, in which the goals of the system to be developed are considered by the technical staff, the ultimate users, and the management from whose budget the system is being paid for. The notation and the level of formality used to capture the requirements are selected according to the standards enforced by the organization. The second phase is *Systems and software design*, in which the hardware and software process requirements are determined and considered in relation to the overall IT architecture that the system is to be incorporated within. The software design aspect leads to

the specification and design of each individual software component, and again the degree of formality of specification is determined by the organization and its policies. Stage three is *Implementation and unit testing*, in which the programs are coded and tested as individual entities. Stage four is *Integration and system testing*, in which the individual programs are tested as complete systems. The final phase is the *Operational and maintenance* phase, generally the longest phase of any software development, involving modifying the code, continued testing, and integration with other software programs.

The waterfall model was originally considered a one-pass development, with each stage being undertaken once. However, it was soon realized that, when an error was found or a modification was required in any stage, it was important to return to earlier stages and amend the design there, so that all stages would reflect the same model of the system. Hence the waterfall model is sometimes referred to as the *Iterative* waterfall model.

Business value proposition

The waterfall model has a long history and many organizations have incorporated a version of it into their application development model. Many other techniques, such as cost modeling systems and project management systems, have been built around the waterfall model and provide an established methodology for software development.

Summary of positive issues

The model is an established approach to software development; tools and techniques associated with the waterfall model have been developed and there is a wide user base.

Summary of potentially negative issues

The approach generally lacks formality and is not a methodology that would achieve a high-level ranking on the Software Engineering Institute's (SEI) Software Capability Maturity Model (CMM). The waterfall model in its non-iterative version does not reveal problems until the end of the lifecycle, the point at which the cost to fix the error is highest. The iterative model does facilitate earlier detection of errors but requires an update revision of the documents and processes at each level, a task that is frequently underperformed or not performed at all.

References

- http://www.sei.cmu.edu/sei-home.html.
- I. Sommerville (2004). *Software Engineering* (New York, Addison-Wesley).

Associated terminology: Formal methods.

Web services

Foundation concept: World Wide Web.
Definition

1. Any service provided by a web server and accessed through a web browser.
2. A particular protocol or group of protocols used by applications to exchange data over a network.

Overview

This is one of those troublesome phrases that for many years had a perfectly obvious and straightforward meaning (a "web service" is any service provided via the World Wide Web), but was later kidnapped and used for a different meaning by a small but demanding subset of society. Since the two meanings are closely related, both being kinds of internet software technology, but very different, it can be hard to tell which meaning any given speaker intends.

1: Any service provided through the World Wide Web, in other words anything that can be accessed over the internet through a web browser, is a *Web service*. This covers a great multitude of things, from simple static web pages, through information services, digital music and movies, to

e-commerce and complex interactive environments.

2: When applications need to interact or share data across the internet without human assistance, some common language and protocol of communication must be provided. *Web services* refers to the entire collection of data languages and communications protocols used by a group of applications for this purpose. Most versions of web services insist that data is represented using the *XML* language. Various protocols are available, including the existing standards *FTP* and *HTTP*, and the special-purpose *SOAP* (Simple Object Access Protocol) and *CORBA* (Common Object Request Broker Architecture). Additional systems such as *WSDL* (Web Services Description Language) and *UDDI* (Universal Description Discovery and Integration) are available to allow systems and applications to identify themselves and others, and to communicate how a particular web service is to be used.

Business value proposition

The development of internet and web-based technologies has resulted in the development of many different models through which businesses can interact and communicate.

The most basic form of "web service" is typically any interaction with a remote web page, such as viewing its contents, using the web site to order a product, accessing a database, or other function. The sophistication of this form of web service leads to the creation of e-commerce business models through which commerce is transacted.

A second form of web service, sometimes referred to as a "web-enabled application," is just that, an application accessible through the use of internet technologies and through which data and/or services can be accessed. For example, an employee who travels extensively on company business may remotely connect through the internet with their company's systems and use a variety of applications. Web-enabled applications have been embraced by many organizations and government entities, and provide a flexible approach to systems access. For example, The US Navy–Marine Internet initiative, commenced in 2000, aims to allow over 300 000 users access to all of the US Navy's estimated 80 000 applications over the internet.

The technologies that underlie web services also facilitated the creation of business-to-business e-commerce and data exchanges, including the provision of data synchronization engines and data pools.

Summary of positive issues

Web technologies continue to mature and their development allows greater connectivity between entities. Web services allow access to remote applications.

Summary of potentially negative issues

The terminology associated with web services is used ambiguously. Web services can be technically demanding to implement.

Reference

- H. Deitel, P. Deitel, B. DuWaldt, and L. Trees (2002). *Web Services: A Technical Introduction* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Data pool, Internet, Protocol.

Wireless Application Protocol (WAP)

Foundation concept: Wireless network.
Definition: The Wireless Application Protocol is an open-specification communications protocol for providing information services on wireless devices.
Not to be confused with: WAP – wireless access point.

Overview

The *Wireless Application Protocol* (WAP) evolved out of a need for wireless providers to create a common platform through which their devices could inter-operate. WAP 1.0, which was developed under the auspices

of the *WAP Forum*, was released in 1998 and was subsequently developed into the backwards-compatible WAP 2.0, released in January 2002.

WAP provides an environment in which web applications can run on hand-held devices using *Micro-browsers* (browsers designed to operate within the limited environment of a very small computer). As such, WAP is based on a six-layer protocol stack similar to the OSI standard *Seven-layer model*, and has been developed to support internet protocols including TCP and HTTP. Central to these layers is the Wireless Application Environment Layer which provides the application-developer interface through which developers can write Extensible Hypertext Markup Language Mobile Profile (XHTMLMP) documents that may be displayed on any device with a WAP micro-browser. WAP's *WAE* (Wireless Application Environment) also facilitates the development of more advanced "presentations" through other standards including WML (Wireless Markup Language), WML Script, and WBMP (Wireless Bitmap image).

The basis of WAP operations is the standard client–server model, in which a request for service is sent over the network to the appropriate server identified by its IP address or URL. In this case, communications are wireless-oriented; the request commences as a wireless signal, and is responded to by the web server and sent back across the network.

The WAP system is capable of running on top of any operating system, including those specially written for hand-held devices with small amounts of memory and limited processing capabilities (these operating systems are often optimized, slimmed-down versions of the full operating system), as well as being optimized to maximize the relatively low data transfer rates associated with wireless systems.

Closely associated with WAP is *WML* (the Wireless Markup Language). WML is very closely related to HTML, the standard language for web pages, and is compatible with the XML standard, but is optimized for wireless use. Instead of using web *pages*, WML treats data for display as a *Deck* of *Cards*, each card containing a small amount of data, with the user expected to navigate through the deck at their own pace. Many wireless devices have severely limited display regions (such as the small screen on a cellular phone), and could not be used to view normal web pages. A common practice is to provide a *WAP gateway*, acting as a proxy between servers and viewers, converting standard HTML to a reduced-size WML document for convenient viewing.

Business value proposition

The WAP protocol was a major breakthrough in mobile computing since it unified vendors under a single set of protocols and avoided the frequent problem of a "land-grab" mentality under which different proprietary protocols battle it out for market dominance. WAP provides a stable protocol environment through which developers can create web presentations aimed at devices with micro-browsers, including PDAs, cell phones, and two-way radios.

WAP must be assessed from a security perspective, in that, when the system's full capabilities for strong encryption (similar to the *Secure sockets layer*) are used together with a secure server gateway at the recipient's side of a request, the security of the data is high. However, there is a possibility of using a weakened encryption system and disabling the security system. For organizations or even individuals using WAP-enabled devices to transfer sensitive data, careful consideration of end-to-end security needs to be made by a professional systems manager.

Summary of positive issues

WAP is an open standard supported by the WAP Forum. The standard is universal

in hand-held devices that support micro-browsers. WAP allows developers to create presentations specifically for micro-browsers. WAP supports a form of *Digital certificates* and other security enhancements.

Summary of potentially negative issues

WAP offers a development and presentation environment that is limited by processing hardware and memory, and by the data transfer rates of the wireless system. WAP has some security concerns associated with the *WTLS* (Wireless Transport Layer Security) protocol which encrypts the data transmitted, and governs the security at the WAP gateway where the server connects to the network.

References

- www.wapforum.org.
- C. Arehart and N. Chidambaram (eds.) (2000). *Professional WAP* (Chicago, IL, Wrox Press).

Associated terminology: Network, LAN, Protocol, HTML, XML, Security, Digital certificate.

Wireless network

Foundation concepts: *Network*, *LAN*.
Definition: A local area network in which the connections between computers use not wires, but radio-frequency broadcasts.

Overview

Electrical signals naturally flow along metal wires; they require no special transmitters or receivers, and are exceptionally cheap and reliable. They are also highly directional: a signal inserted at one end of a wire flows straight to the other end, and does not leak out in unwanted directions. This makes metal wires seem like the ideal medium for inter-computer data connections, and indeed most network connections are in the form of normal electrical cables.

Metal wires are not perfect; they get in the way, trip people up, and get broken and tangled. If the public is to be given access to a network, connectors and sockets have to be provided, and the public will quickly break them or fill them with foreign substances. Wires also prevent mobility; it is hard to move around freely while using a computer that is connected by a cord to the wall.

When mobility or public access is required, or under other circumstances when wires may be undesirable, a wireless network is of course the solution. On a wireless network, every device must be fitted with a transceiver (a transmitter and receiver in one) that converts between the native electronic signals of the computing device and the radio-frequency transmissions used by the wireless network. This additional equipment adds some small expense, but at least there is a single universally adopted standard, or at least a family of mutually compatible standards (in the IEEE 803 suite), so wireless network adapters are as standard as the ethernet adapters that they replace. Unless a wireless network is to be totally isolated and not part of the internet, there must also be a *Base station* or *Wireless access point*, to provide a gateway.

Wireless networks are not well suited to all circumstances. Wireless signals are not easily contained or directed. When two networks are close together, great confusion can be caused for transceivers in the area of overlap. If a base station is not properly secured, unauthorized users can "hitch a free ride" on the network, using an internet connection that someone else has paid for. Most wireless network base stations are easy to secure against unauthorized use, but it does require some user configuration, which is most often skipped. If any network traffic is not encrypted, it can easily be intercepted by an undetectable interloper. Inside buildings, walls and equipment, and even moving people, can interfere with network signals, and

cause unpredictable weak spots where there is no network access.

Business value proposition

Wireless networks provide organizations with the ability to maintain or even increase the connectivity between computers whilst reducing the complexity involved. Wireless network systems allow a variety of system solutions to be created within an organization, including so-called wireless "hot spots" where a transceiver allows wireless connections to be made by any authorized device. While wireless networks enable employees to work anywhere there is a hot spot, many organizations are structured so that employees work in fixed locations or can access their data anywhere within an organization by logging onto a fixed cable-connected networked computer. While the benefits can easily be determined for an institution such as a university where the "customers" (the students) are working from a variety of locations in a day, or a hotel where the guests require internet access as a service within "public areas," the cost justifications in developing a business case for a corporate move to wireless networks may be more difficult in the case of a traditional static workforce. The case for equipping a company's mobile workforce with wireless technology can be based upon productivity gains, but must be balanced with the risk of security breaches.

Wireless network use goes beyond the simple connection of laptops to corporate networks, and includes the development of devices such as RFID tags that allow the monitoring of devices or items within a wireless coverage zone. Beyond the IEEE 803 standards-based technologies there are other mechanisms through which wireless communications can occur. One such technology is the wireless cellular network that supports the 3G computing devices embedded in telephones.

Summary of positive issues

Wireless networks provide computer users and computing devices with flexibility in terms of their operational locations. The standards are well defined both at the IEEE level and at public telephone network level. Many wireless networks in public spaces are provided as a "free" service and some municipalities are providing or contemplating the provision of wireless networks to their communities. RFID and other technologies use wireless network technologies.

Summary of potentially negative issues

Wireless network systems are slower than cable-based systems. The cost of providing wireless networks needs a solid business case to be built around it. For some organizations such as coffee shops, it may be a positive in attracting customers but a large negative if the customers don't buy coffee and just linger for free network connections in a nice atmosphere. The use of an insecure wireless system is detrimental to any organization's networks and intellectual properties.

Reference

- K. Pahlavan and P. Krishnamurthy (2001). *Principles of Wireless Networks: A Unified Approach* (Englewood Cliffs, NJ, Prentice-Hall).

Associated terminology: Network, Local area network, Wireless Application Protocol.

World Wide Web

Foundation concepts: Internet, Hypertext.

Overview

Although the term "World Wide Web" is often taken as a synonym for "the internet," there is a subtle shade of difference in meanings. "The internet" really

describes the worldwide collection of interconnected computers, the hardware that connects them, and the fundamental software and protocols that make communications between them possible. The "World Wide Web" refers to the web of interconnected documents, data, and applications that spans the internet.

The internet existed long before the World Wide Web that we know today (see *Internet* for details). Since the early days of ARPANET, it has been possible for anyone with internet connectivity to put any collection of data online, so that anyone else with the same connectivity may download and access them at will ("A file transfer protocol" was published by Abhay Bhushan of MIT in 1971). However, those data files were stand-alone documents: each could be viewed in any convenient way, and they may contain references to other online documents, but those references would be simple inert text as might be printed in a book. Any reader wishing to view a referenced document would have to find it, manually download it, and separately view it; not necessarily a difficult process, but not the smooth operation we are familiar with today.

The World Wide Web did not exist until the introduction of a practical *Hypertext* system by Tim Berners-Lee of CERN (*Conseil Conseil Européen pour la Recherche Nucléaire* or the European Council for Nuclear Research) in 1990. This included *HTML* (the HyperText Markup Language) and *HTTP* (the HyperText Transfer Protocol). HTML (see *Hypertext* for details) is a simple language that allows the creation of documents that mix text, graphics, and other media, which includes *Hyperlinks*: references to other online documents that can be retrieved and displayed automatically, using the now familiar point-and-click technique. HTTP is the even simpler language that document display software (web browsers) and the remote document storage systems (web servers) use to communicate requests and their corresponding documents.

It is HTML and HTTP that make the seamless browsing of the web, or surfing of the net, possible. HTML and HTTP together allow the carefully designed graphical layouts with an almost infinite variety of fonts, text sizes, backgrounds, colors, images, and other graphical elements, and the online forms and their automatic processing on which e-commerce is founded. Of course, many other applications provide the same mixture of text and graphical elements, and many provide better control over the appearance of a page, but the fundamental thing that makes the web more than just a collection of files accessible through the internet is the ability to follow a reference to another online document anywhere in the world instantly and automatically. That is the difference between text and hypertext; it is also the difference between the internet and the World Wide Web.

In order to publish documents on the web, only one thing is required: a *Web server*. A web server is usually thought of as a networked computer with this designated task, but it is actually just a simple piece of software running on a networked computer. A single computer may in fact support a large number of different web servers. Web server software compatible with all popular computers and operating systems is freely available for download, but there are also commercial versions available that may provide more support. Once a web server has been installed and started on a networked computer, it is configured to make certain groups of files available for browsing, and it simply waits for requests composed in the HTTP language. Each request received is analyzed to discover which actual file or application is wanted, and, if that access is permitted, the results are transmitted back. The selection of files made available by a single web

server is usually referred to as a *Web site*, although this term is not strictly defined.

For a web server to be accessible, its IP address must be made known. The most common way is through DNS, the *Domain name service*. This is a system that associates humanly understandable and memorable names (such as "YourCompanyName.com") with the totally unmemorable and unfriendly numeric IP addresses (such as "192.168.31.108") actually used in internet communications (see *Domain name* for details). Given just a domain name or an IP address, web clients or browser applications know exactly how to contact the indicated web server, and request an appropriate starting document.

Normally, each web site has its own *Home page*, a specially designed hypertext document (usually stored in a file called "index.html") that serves as an introduction and guide to the whole site. When a web browser is given only a domain name or IP address, it asks the associated web server for its home page file, and displays that to the user. The home page would normally be expected to contain hyperlinks to other documents on that site or others, enabling visitors to find whatever they want.

The term *Web browser* usually refers to the software application (Mosaic, Netscape, Internet Explorer, etc.) through which a human user accesses web content, although sometimes the term is applied instead to the human user. It is the browser application's responsibility to take web addresses entered by the user and convert them into proper HTTP requests sent to the appropriate server, to display the response appropriately, and monitor mouse clicks, converting them into additional HTTP requests when those clicks occur over hyperlinks.

Domain names refer to whole web sites. Web addresses entered by users, or encapsulated in hyperlinks, may instead refer to individual documents on a particular web site. A reference to a file, document, or application on a web site is encoded as a URL, which includes both a domain name and additional file designations (see *URL* for details). URLs are the basic address designators for the World Wide Web, just as IP addresses are for the internet.

Recently, two newer models for web-based information dissemination have become popular, the *Blog* and the *Wiki*. A blog (web log) is little more than an online public diary: a person writes thoughts, opinions, poems, or whatever else comes to mind, perhaps adding pictures and audio components, to a continuously evolving personal web site in the hope that others will read it. Bloggers usually arrange to have links to their sites placed on blog-directory sites so that others will find them. Most blogs are like most other self-published work, of no interest to anyone but the author, but a few of the more talented authors do gain quite a following.

A wiki (derived from a Hawaiian word) is a public web site at which anyone in the internet community may add content or even edit existing content. Occasionally there may be a designated moderator who checks all changes and additions, but most commonly there is no oversight. The best known of all wiki sites, Wikipedia (http://en.wikipedia.org/wiki/Main_Page), is growing into a large and sometimes useful free online encyclopedia. Anyone may contribute articles, but, since there are a great many frequent users, untruthful or malicious additions are usually caught quite quickly, although vandalism of its content is, of course, a recurring problem. Wiki sites may be used as general meeting places for people to share opinions, as bulletin boards for posting notices, or as the center of a collaborative online workforce.

Business value proposition

The World Wide Web is a medium for the distribution of information. It allows

individuals, organizations, and governments to access data and to make their own data accessible very cheaply. The development of an externally facing corporate web site is usually centered on one of several business models, including business-to-customer (B2C) commerce, business-to-business (B2B) commerce, business-to-government (B2G) commerce, government-to-citizen (G2C) commerce, online-community models, and blogs. It is also possible to use web technologies to provide web sites for internal corporate use only; these would house such things as internal corporate training materials, data on human resources, and documents relating to technology and research. This provides a low-cost model for the dissemination of information and helps companies to control and monitor access to resources.

The development of externally facing corporate web sites requires careful consideration not only of the design of the hypertext but also in terms of the intent of the web site as related to the corporate strategy. This requires achieving a balance amongst the market demands that the site aims to address, the service level to be provided by the site, and the brand implications of the site and the technology that supports it. A key aspect of the brand is the selection of the domain name, since a memorable name such as "YourCompanyName.com" is clearly easier for customers to remember than an IP addresses such as "192.168.31.108." Companies need to determine whether they wish to use the same name as their physical brand name or whether the online brand is going to have a separate identity.

The deployment of web sites on a corporate web server also requires an assessment of the impact of that sales channel upon the corporate value chain, as well as an assessment of the scalability and robustness of the technology behind the site. For example, a stress test to assess the system under peak demands would be valuable in order to avoid crashes at true peaks in systems demand such as Christmas shopping periods, and for stock trading at key IPOs. There is a very large array of tools and technologies available to assist web masters in the development, testing, and deployment of web sites.

The choice of browser to access and read web sites has evolved since the "browser wars" of the 1990s. While there are dominant vendors in some computer categories, there are many emergent browsers for several classes of computer. Free open-source browsers are available for desktop systems and many vendors offer micro-browsers for hand-held devices such as cell phones.

The advent of blogs has required many corporations to develop a formal policy on employees' online behavior, and strictly enforce it. Policy usually includes the obligations of the blogger toward the company, and requires that blogs relating to any company activity or information be authorized by the company's media relations department, and include the author's real name and their official role in the company. The policy document should stress the need to maintain corporate intellectual property and that blogs should not bring the company into disrepute by including personal opinions on sensitive subject matters, such as on an IPO during its quiet period. Blogs can be positive and many executives and business personalities who blog have a large following. The blog can act as a vehicle for expressing intellectual content and building an online community, ultimately adding value back to the company.

The development of wikis has proved a useful mechanism for collaborative efforts. Within corporations, wiki web sites allow for a wide variety of collaborative work spaces, such as the joint development of documents, product plans, and specifications. The relatively low cost of maintaining wiki sites combined with their easy

access through a standard browser makes this form of collaboration tool a medium through which remote or distributed workers may contribute to a project in an asynchronous manner.

Summary of positive issues

The World Wide Web provides a popular mechanism for disseminating information across the internet. A web site can be internal to an organization, whereby content may be restricted to a workforce or group of workers. Externally facing web sites can be used to provide information or to sell products. The World Wide Web is based upon a set of open standards and protocols that continue to evolve to meet new challenges. The technologies and services associated with web development are widely available and have a wide range of capabilities; many tools are open-source and free to download. Domain names can be acquired to reflect the branding statement of the organization or individual.

Summary of potentially negative issues

The technology underpinning the World Wide Web continues to evolve and requires that the corporate or individual web master maintains a working knowledge of advances. Popular domain names may already have been acquired by others. Not all browsers have the same capabilities: micro-browsers on hand-held devices do not have the same capabilities as desktop computer-based browsers.

Associated terminology: Internet, Server–client, Hypertext, Internet protocol.

WYSIWYG

Definition: An acronym for "What You See Is What You Get," describing the visible form of a document while it is being developed. If the visible form the document takes while it is being created is the same as one produced on an output device such as a printer or through a browser then the document production system is a WYSIWYG system.

Overview

Word processors and other content-creation software may be classified as either WYSIWYG (pronounced whiz-ee-wig) or non-WYSIWYG. Word processors in the 1970s and early 1980s such as WordStar generally operated in one of two modes: *Edit mode* and *View mode*. When in edit mode, strange annotations would appear in the text (like the HTML markup tags of today, but far more arcane); users could edit the text, but it would often be hard to visualize how it would appear when printed. In view mode, the document would appear in an approximation of its final printed or display form, with roughly correct text layout, and perhaps even the right fonts, but it would not be possible to make any modifications to the document without switching back to edit mode. These were non-WYSIWYG systems. Under a WYSIWYG system there is only one mode of operation: editing is performed while viewing the document in its production form, and the document is shown in as close to its final form as the display hardware permits.

The first WYSIWYG word processor, *BRAVO*, was created at Xerox PARC in 1974 and this style of word processing has continued to grow in use, having been popularized by Microsoft's *Word* program and Adobe's *Frame-Maker*. However, development of markup-language-based document preparation systems continues and systems such as LaTeX facilitate the production of high-quality documents and publications, especially when specialized notations and symbolic forms are required. Often a document processing system will provide two interfaces for working in HTML and similar markup languages, allowing the user to directly edit the markup tags, or to work in

WYSIWYG mode, indirectly modifying the document's final form.

Business value proposition

The use of WYSIWYG environments is intuitive; they frequently require less user training and can speed up the development of documents and projects. Non-WYSIWYG document preparation systems and environments can provide superior control over the documents produced, and may even be easier to use when specialized document formats are to be developed. Environments that facilitate markup languages such as HTML can also reduce the complexity of development.

Summary of positive issues

WYSIWYG systems are easy to use, intuitive, and may require less training than non-WYSIWYG systems. WYSIWYG word processing systems have achieved widespread adoption.

Summary of potentially negative issues

The WYSIWYG systems are more expensive, may reduce direct control over the documents being produced, and may lack the resources available to non-WYSIWYG developers.

Associated terminology: Hypertext.

X.12

Definition: An ANSI protocol that defines a set of standards for documents involved in electronic data interchange.

Overview

The American National Standards Institute (ANSI) chartered an Accredited Standards Committee (ASC) in 1979 to develop a set of standards for the documents that are exchanged through *Electronic data interchange* (EDI). The standard has periodically been revised by the ASC, with version 2 being established in 1986, version 3 in 1992, version 4 in 1997, and version 5 in 2004. The standard defines *Transaction sets* to capture the information that would be found in a paper document. For example, the X.12 transaction set 850 represents an outbound purchase order, 860 a purchase order change document, 855 a purchase order acknowledgement document, set 856 represents packing slips, and 810 represents invoices.

The X.12 standard breaks a transaction set into logical groups of data called *Segments*. For example, the 850 purchase order set could contain the following segments: a transaction set header, an initial segment, a reference number, a date/time reference, carrier details, name, product identifier, physical details of the item, marking/packing/loading, destination quantity, and transaction totals. All segments for each transaction set are extensive because they have to be usable by many different organizations, and it is usual for only subsets of the defined segments to be utilized by each company. The segments are themselves composed of data items; thus the item "marking/packing/loading" could contain the fields item description type, packaging characteristic code, association qualifier code, and packaging description code. In order for the two companies that are exchanging the data to understand which codes are to be used in each transaction, a *Mapping document*, which specifies exactly which segments, data items, and fields are to be used, is agreed upon and shared by the two parties.

Business value proposition

The X.12 standard is widely used and has undergone consistent revision and modernization since its creation in 1979. The standard covers a very wide range of documents, with approximately 300 transaction sets having been developed.

The X.12 standard continues to evolve to meet the demands of practitioners. Version 4010 onwards is year-2000 compliant, and addresses new regulatory requirements such as the 1996 Health Insurance Portability and Accountability Act (HIPAA), which in the United States requires that all healthcare providers be X.12-compliant by August 1, 2003.

The use of the X.12 standard for EDI provides companies with the opportunity to reduce transaction costs, speed data exchange, increase reliability, and simplify procedures.

Summary of positive issues

The standard is mature, having undergone more than 25 years of development. It has been adopted by a wide variety of industries and by more than 300 000 companies worldwide. The standard has continued to evolve and is a required federal standard in the United States for healthcare providers.

Summary of potentially negative issues

The three major EDI standards, namely the United Nations Guidelines for Trade Data Interchange (UNTDI), the United Nations rules for Electronic Data Interchange for Administration, Commerce and Transport (EDIFACT), and ANSI X.12, use transaction sets that are not compatible and need middleware systems to translate the data from one format to another. The ASC commenced work in 2001 to make X.12 and EDIFACT share common transaction sets

and move toward a global standard, since X.12 is widely used in the United States and EDIFACT is widely used in the European Union. The ASC X.12 Standards organization also began an initiative in 2001 to harmonize X.12 with XML-based messaging standards, but XML and X.12 remain separate approaches to document messaging.

Reference

- http://www.x12.org/x12org/index.cfm.

Associated terminology: XML.

X.25

Foundation concepts: Packet switching, Network.
Definition: X.25 is an ITU-T protocol that defines a set of standards for connecting user and network devices.

Overview

The X.25 standard was defined by the International Telecommunications Union (ITU), an organization that operates under the auspices of the United Nations to develop and coordinate global telecommunications standards. The ITU-T (the final "T" signifying the telecommunications sector group) uses a naming convention of a letter and a number separated by a period. The letter "X" signifies a group of related standards in a common area, namely data networks and open systems communications. The number "25" signifies that the standard covers the area of packet switching.

The X.25 standard applies to the first three layers (network, data, and physical) of the OSI *Seven-layer model* and is typically used on the networks of telecommunications companies (generally known as common carriers), and enables user devices and network devices to communicate with each other regardless of their type.

Communication over an X.25 network begins when a data terminal such as a personal computer calls another data terminal to request a joint communication session over the network. If the called terminal accepts the call, a full duplex (bi-directional) transfer of data can occur. The communication session can be terminated by either party at any time.

Business value proposition

The X.25 protocol was initially developed in 1976 to facilitate the creation of wide area networks over public telecommunications equipment. The technology is mature, having been revised in 1980, 1984, and 1988, resulting in a debugged and stable environment that provides high-quality data connectivity. X.25 has been implemented widely and products with X.25 certification are widely available.

Summary of positive issues

The technology is mature, stable, and provides high-quality connections. Products adhering to the standard are widely available. X.25 applies to dial-up connection methods.

Summary of potentially negative issues

The X.25 protocol incurs a turn-around delay on messaging between terminal devices. This can be detrimental to heavy bi-directional communication over an X.25 network. Line speeds of 64 kbps are too slow for many applications that require higher levels of bandwidth to perform their tasks usefully.

References

- http://www.itu.int/home/.
- M. Clark (2001). *Networks and Telcommunications* (New York, John Wiley and Sons).

XML

Definition: The Extensible Markup Language (XML) is a standard language for describing and encapsulating data.

Overview

The concept of an Extensible Markup Language (XML) took shape in November 1996, evolving from the Standard Generalized Markup Language (SGML) which became ISO 8879. The major concept underlying markup languages is that they contain tags or markers that signify some form of coding. In the example

```
<bold> word </bold>
```

the tags <bold> and </bold> signify that whatever comes between them should be printed in **bold**; mechanisms of this nature were employed by many early word processing systems. This concept is employed by a variety of markup languages, most famously HTML, which is used to mark up data so that the format of the data can be controlled when displayed by a web browser. While the tags in HTML are predefined so that all systems use the same tag sets and their meanings are universally known, the tags in XML are not predefined. This allows (and requires) developers to create their own data structures unique to a task's requirements. This is useful when organizations wish to exchange data and choose not to follow the strictly defined regimen of X.12. For example, the X.12 transaction set 850 for purchase orders could be replaced by a purpose-designed purchase order schema in XML:

```
<?xml version="1.0"?>
<purchase_order>
<description>
  <order_type>
    "Urgent Shipment"</order_type>
<Destination_Address>
  "Head Office"</Destination_Address>
<Item_Description>
  "Large Widget"</Item_Description>
</description>
</purchase_order>
```

The names and placements of the tags are purely at the discretion of the individual designer. The structural details of the tags, i.e., the element types, their attributes, and so on, are defined by an XML schema (itself written in XML) that is transmitted to the partner system in a data exchange prior to the data transmission itself. However, another group of companies may use a completely different set of tags to denote purchase orders to suit their particular needs. The flexibility of XML allows easy adaptation to individual requirements, but this may require additional software to translate between different XML schema designs.

Business value proposition

XML was developed under the auspices of the World Wide Web consortium (W3C.org) and the W3C XML working groups continue to develop the language and its extensions, providing recommendations, working documents, and documentation. XML is platform-independent and widely supported, with open-source and proprietary parsers available, and XML technologies have been embraced by a wide array of software vendors from specialist XML vendors to ERP vendors such as SAP. The technology has continued to grow in use, and a registry through which companies can automatically discover and connect with each other via XML, known as the *Universal Description, Discovery and Integration* (UDDI) initiative, has been created.

Summary of positive issues

The XML approach to data description transfer is flexible, cross-platform, and widely utilized by software vendors. The language is supported by W3C and continues to evolve, supporting encryption, communicating with web services, and facilitating end-to-end transactions.

Summary of potentially negative issues

XML may not be used for the transmission of healthcare data in the United States, because HIPAA mandates the use of X.12N

(The N signifies "insurance" documentation under the X.12 EDI protocol) for dental, professional, and institutional transactions, and the National Council for Prescription Drug Programs (NCPDP) standards for pharmacy transactions. Security when using XML has to be considered and a *Secure sockets layer* (SSL) approach is frequently adopted for securing XML data transfers.

Reference

- http://www.w3c.org.

Associated terminology: X.12, Hypertext, Protocol.

Y2K problem

Definition: The problem caused by using two digits to represent the year in recorded dates (e.g., 97 to mean 1997), and by extension any problem caused by using a data-limited representation based on invalid assumptions about the possible range of values.

Overview

The "year-2000" problem, or Y2K, is very well known. Many years ago, high-capacity long-term storage was expensive, and by modern standards not very high-capacity at all. Every commercial programmer knew the need to compress data so that more effective use could be made of the limited capacity available. In the mid 1960s, programmers working on commercial applications never imagined that their products would still be in use 35 years later, and storing the "19" that appeared without fail at the beginning of every year number seemed like an absurd waste.

In all respects but one, they were right. The Y2K problem was not caused by incompetence or laziness amongst programmers. The applications they wrote 35 years ago are generally not still in use. Unfortunately, the data sets that those applications produced, replete with their missing 19s, outlived those applications and have remained in continuous use, growing rapidly, ever since.

Fortunately, the world's programmers successfully updated all (with a very few exceptions) applications and data sets in time, and the financial ruin and end of the world as we know it predicted by the popular press was averted.

Unfortunately, very similar problems still exist in many guises. The use of seven digits for telephone numbers within large geographical areas was a variant of the Y2K problem: who would ever have expected the nearly ten million telephone numbers in each area code not to be enough? Unix systems represent dates and times by the number of seconds elapsed since midnight on January 1, 1970, but frequently use a numeric format that can count only up to 2 147 483 647. Two billion is a large number, but two billion seconds is less than 70 years; many Unix systems have their own version of the Y2K problem scheduled for some time in January 2038.

> Those who do not learn from history are doomed to repeat it. *George Santayana*

> We learn from history that we learn nothing from history. *George Bernard Shaw*

Business value proposition

The Y2K data problem should act as a reminder to all IT managers and programmers that the data format they choose at the outset of a project needs careful consideration since data structures created by their systems may be around for a very long time.

Summary of positive issues

The vendors of software applications today aim to build scalable solutions and provide tools to help organizations migrate their systems from one vendor to another, allowing companies to move from outdated software using proprietary vendors' formats.

Summary of potentially negative issues

The Unix problem and other "time bombs" are still in existence, some known and some still unknown at this time. Software from vendors that either have gone out of business or no longer support their software may contain undetected problems that could surface at any time, at which point there may be no "quick fix." Or worse, the problem could be catastrophic for the organization.

Associated terminology: Reliability, Database.

Zip

Definitions:

1: A popular file compression system.
2: A medium-capacity removable disk format.
3: A data processing operation.

Overviews

(1) A *Zip file* is a single file that may be copied, transmitted, or archived as a single unit, which may contain within it an unlimited number of other files, structured in folders and sub-folders. Depending on their content, these files may be very heavily compressed; this makes zip files a popular form of attachment for emails. Using standard software, the contents of a zip file may be viewed and modified in the familiar "drag-and-drop" windows style, and available application program interfaces allow applications to directly access sub-files from a zip file.

(2) *Zip disk* is the trade name for the most popular variety of removable disks, and has almost become a generic name for that kind of product. They are small devices in protective casings, similar in concept to the floppy disk. There are various varieties of zip disks with different capacity, speed, and price configurations. For a period, zip disks were a very popular medium for data backup and transport, but they have now largely been eclipsed by *USB flash memory* devices.

(3) A *Zip operation* is a simple data processing step, in which two or more equal-length lists of data records are combined into a single list of the same length by merging or concatenating the items from corresponding positions.

Business value proposition

Zip files are a very convenient and universally accepted format for archiving data for backup and for packaging collections of data and applications for electronic transmission.

Zip-like disks filled an important niche in the business and personal computing world, but have been left behind by technological evolution.

Summary of potentially negative issues

The Zip file system works through the use of proprietary software owned by the PKware corporation, or a competing version owned by WinZip International LLC. Until recently the two versions were completely compatible, but with the addition of strong encryption that ceased to be the case. In 2003, the PKware corporation filed for a US patent on the zip specification, which it had previously published openly since its invention in 1986.

Zip disks are also proprietary; the intellectual property rights are owned by the Iomega Corporation.

Associated terminology: Compression, Disk, Storage, Database.

Index