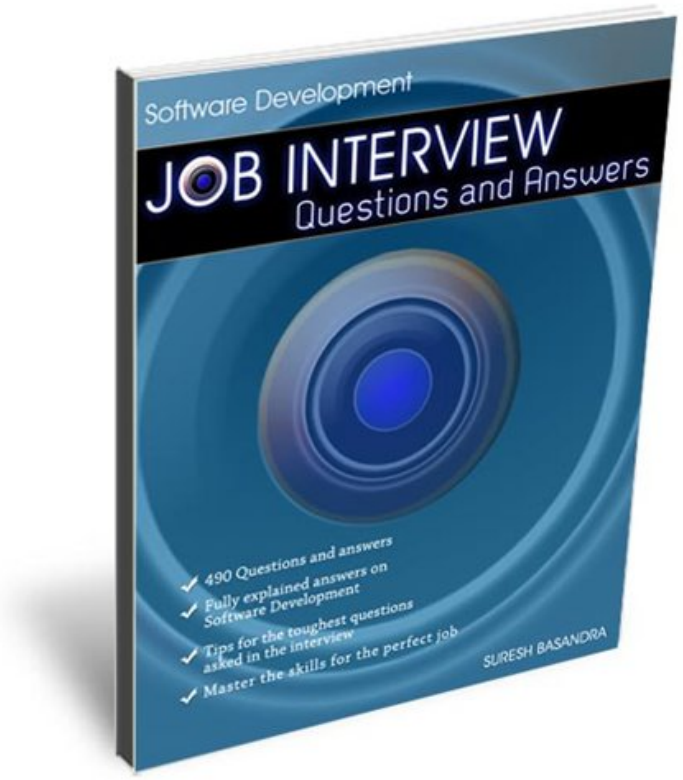
Software Architecture and Design Questions and Answers



Suresh Basandra

Software Architecture and Design Questions and Answers

PDF ISBN-10:
PDF ISBN-13:
EPUB ISBN-13:

Current Edition: 2011
First Edition: September 2010

Published and typeset by Basandra Books, California, USA
Designer: Karuna Basandra
Webmaster: Gagan Basandra
Editor: Kriti Basandra
Web: http://books.basandra.com/

Preface

A few years ago we could basically pick and choose the job we wanted. If we were dissatisfied we moved on. Job mobility was the order of the day. Now, in far too many cases, outsourcing, downsizing, and/or rightsizing seem to be the order of the day. Job stability often appears to be little more than words found in a dictionary.

Here is a book that will enable you to beat out the competition. Your competition is all those people trying to get the same job you are trying to get. Let us face it, as far as you are concerned there is only one person that is going to get each job. As for the job you want, you want it to be you, not them. If you play your cards right, try as hard as you can, prepare yourself, and never give up, you will succeed. And this is what this book is all about … preparing and succeeding.

There are numerous FAQs, questions and answers, articles on Web, books, etc. on the market covering almost every conceivable topic, however, from an interview perspective it is not feasible to brush up on all these materials when you may not have much time. With technologies evolving so rapidly, it is hard for even the most knowledgeable professional to handle job interview with confidence and precise answers to tough technical and management questions. This is especially true when it comes to information about such vital new areas of job opportunity as software engineering, etc. Here is a book that will enable you to ace any technical interview as it provides answers to challenging and difficult questions in a wide range of important areas. The key objective of this book is to cover all the core concepts and areas which all software developers, designers, administrators, and architects should be familiar with to perform in their jobs or to do well in interviews. The interviewer can also use this book to ensure that they hire the right candidate depending on their requirements.

Software Architecture and Design Questions and Answers is a one-stop reference that both beginning and experienced software engineers will use successfully in technical job interviews and countless on-the-job situations. Providing the answers to critical questions, from the simplest to the most advanced, this book is arranged to get you the information you need the moment you need it. You will find helpful explanations of crucial software development issues.

Packed with numerous real-life examples and case studies, Software Architecture and Design Questions and Answers is an indispensable guide covering software architecture and design areas. It is a practical and constantly usable resource for understanding fundamental software development issues and implementing workable solutions based on author's more than 30 years development and management experience.

There are in all about 300 questions with detailed answers so you do not have to worry about spending countless hours browsing the Web or text books for correct answers that get you succeed. This book is not a multiple choice question and answer book as we believe we need to have detailed understanding of concepts and principles to better do our job as a software engineer, administrator, or architect. This book also does not contain any non-relevant material to avoid making it bulky.

Key topics covered include: Software Development, Architecture, Design, Performance, Scalability, Reliability, Object Oriented Analysis and Design, Object Oriented Programming, Aspect Oriented Programming, Software Estimation, Framework, Software Engineering Best Practices, Design Patterns, Service Oriented Architecture, Object Modeling, Data Modeling, Web Services, Web Architecture, Performance Tuning, Installation and Configuration, etc.

In the latest edition, we have demonstrated a commitment to clear, direct writing, with questions that are quick, to the point and stress the core essentials of software development. Those who want to learn more about the profession, as well as those who want to fine-tune their development skills, will find it to be a straightforward question and answer format with hundreds of questions covering key software engineering knowledge areas. Whether you are a seasoned professional, novice, student, or instructor of software engineering, you will appreciate this book for its rich content and ability to test your skill and knowledge.

Simply return the book if you are not 100% satisfied - no questions asked. However, I would request you to please provide your valuable suggestions for improvement since I strongly believe in continuous improvement.

More than 200,000 copies of author's previous books have been sold. Please review author's profile at: http://books.basandra.com/aboutus.html.

It is planned to keep this book updated on a regular basis so that the readers are kept current with latest trends.

Basandra Books introduced its first question and answer book in 2010 and, as the software engineering profession continues to mature, improvements to this book have evolved naturally from various feedbacks on the previous editions. I sincerely appreciate the invaluable feedback provided by various readers and reviewers. Please provide any feedback and corrections using http://books.basandra.com/contactus.html or email to books-support@basandra.com. Thank you in advance for your feedback.

All in all a lot of effort by the author has been put into providing you, the technical and management professional whose livelihood depends on the currency of his or her technical and management knowledge and skills, with as much new meaningful information as possible. If I am able to give you any of the information that you need to survive and succeed then my effort would be considered more than worthwhile.

Suresh Basandra,
California, USA

1.9. Do you typically generate alternative designs?
1.10. How do you choose between these designs?
1.11. Tell me about the last major design you did. What was the purpose? What happened?
1.12. If you had to design for our product (talk about your product here), what approach would you take?
1.13. What are the productivity measurement considerations?
1.14. How do you define productivity?
1.15. What is system architecture?
1.16. What is most critical activity in software development life cycle?
1.17. What are the steps in defining architecture?
1.18. Why do you need architecture?
1.19. Why do you need a layered architecture? Why are layered architectures so useful?
1.20. How do you document or present architecture?
1.21. How do you sell architecture to senior management?
1.22. How do you create an architectural vision?
1.23. Why do you need enterprise architect?
1.24. What may be the reasons why you don't need software architect?
1.25. What is effective architecture? Who is effective architect?
1.26. Who, when and how should architecture decisions be made?
1.27. What is your definition of a well-designed product?
1.28. What is good design?
1.29. What should a software designer consider in the design and implementation of code?
1.30. How do you go about obtaining requirements from your customer?
1.31. How do you make certain you understand what they want?
1.32. Once you have the requirements, what do you do with them?
1.33. To what level do you document them? Please provide samples of requirements gathering documentation work that you have done.
1.34. What is good code? Or what do you consider to be good code?

1.159. What are consoles?
1.160. What are managers?
1.161. Describe the traffic entering and leaving the corporate network? Email content versus Web content.
1.162. How information leaves the corporate network?
1.163. What is leaving the corporation in email attachments?
1.164. Encrypted versus unencrypted content
1.165. What is safety and security?
1.166. What are the advantages of LDAP?
1.167. What is SSO?
1.168. What types of security technologies have you used?

1.169. What is authentication?
1.170. Hash functions
1.171. Describe what PKI is and how it works.
1.172. What is access control?
1.173. Discuss the differences between symmetric key cryptography and public key cryptography. Give an example of when you would use each.
1.174. If you discover a new cryptography algorithm, should you use it immediately?
1.175. Since FTP is insecure, how can you implement a secure solution that is as easy to use as FTP?
1.176. What is role-based user provisioning?
1.177. What is HTTPS?
1.178. What are the key differences between IPSec and SSL?
1.179. What are passfaces?
1.180. What is data marshaling?
1.181. Discuss concerns and limitations while developing for mobile devices.
1.182. What is distinctive about real-time software?
1.183. What is the difference between Google Android and Windows Mobile?

1. Architecture and Design

1.1. What is software design?

Object oriented techniques, and C++ in particular, seem to be taking the software world by storm. Numerous articles and books have appeared describing how to apply the new techniques. In general, the questions of whether object oriented techniques are just hype have been replaced by questions of how to get the benefits with the least amount of pain. Object oriented techniques have been around for some time, but this exploding popularity seems a bit unusual. Why the sudden interest? All kinds of explanations have been offered. In truth, there is probably no single reason. Probably, a combination of factors has finally reached critical mass and things are taking off. Nevertheless, it seems that C++ itself is a major factor in this latest phase of the software revolution. Again, there are probably a number of reasons why, but I want to suggest an answer from a slightly

different perspective: C++ has become popular because it makes it easier to design software and program at the same time.

If that comment seems a bit unusual, it is deliberate. What I want to do here is take a look at the relationship between programming and software design. I have felt that the software industry collectively misses a subtle point about the difference between developing a software design and what a software design really is. I think there is a profound lesson in the growing popularity of C++ about what we can do to become better software engineers, if only we see it. This lesson is that programming is not about building software; programming is about designing software.

Years ago I was attending a seminar where the question came up of whether software development is an engineering discipline or not. While I do not remember the resulting discussion, I do remember how it catalyzed my own thinking that the software industry has created some false parallels with hardware engineering while missing some perfectly valid parallels. In essence, I concluded that we are not software engineers because we do not realize what a software design really is. I am even more convinced of that today.

The final goal of any engineering activity is the some type of documentation. When a design effort is complete, the design documentation is turned over to the manufacturing team. This is a completely different group with completely different skills from the design team. If the design documents truly represent a complete design, the manufacturing team can proceed to build the product. In fact, they can proceed to build lots of the product, all without any further intervention of the designers. After reviewing the software development life cycle as I understood it, I concluded that the only software documentation that actually seems to satisfy the criteria of an engineering design is the source code listings.

There are probably enough arguments both for and against this premise to fill numerous articles. Here we assume that final source code is the real software design and then examines some of the consequences of that assumption. I may not be able to prove that this point of view is correct, but

I hope to shown that it does explain some of the observed facts of the software industry, including the popularity of C++.

There is one consequence of considering code as software design that completely overwhelms all others. It is so important and so obvious that it is a total blind spot for most software organizations. This is the fact that software is cheap to build. It does not qualify as inexpensive; it is so cheap it is almost free. If source code is a software design, then actually building software is done by compilers and linkers. We often refer to the process of compiling and linking a complete software system as "doing a build". The capital investment in software construction equipment is low - all it really takes is a computer, an editor, a compiler, and a linker. Once a build environment is available, then actually doing a software build just takes a little time. Compiling a 50,000 line C++ program may seem to take forever, but how long would it take to build a hardware system that had a design of the same complexity as 50,000 lines of C++.

Another consequence of considering source code as software design is the fact that a software design is relatively easy to create, at least in the mechanical sense. Writing (i.e., designing) a typical software module of 50 to 100 lines of code is usually only a couple of day's effort (getting it fully debugged is another story). It is tempting to ask if there is any other engineering discipline that can produce designs of such complexity as software in such a short time, but first we have to figure out how to measure and compare complexity. Nevertheless, it is obvious that software designs get very large rather quickly.

Given that software designs are relatively easy to turn out, and essentially free to build, an unsurprising revelation is that software designs tend to be incredibly large and complex. This may seem obvious but the magnitude of the problem is often ignored. School projects often end up being several thousand lines of code. There are software products with 10,000 line designs that are given away by their designers. We have long since passed the point where simple software is of much interest. Typical commercial software products have designs that consist of hundreds of thousands of lines. Many software designs run into the millions. Additionally, software designs are almost always constantly evolving. While the current design

may only be a few thousand lines of code, many times that may actually have been written over the life of the product.

While there are certainly examples of hardware designs that are arguably as complex as software designs, note two facts about modern hardware. One, complex hardware engineering efforts are not always as free of bugs as software critics would have us believe. Major microprocessors have been shipped with errors in their logic, bridges collapsed, dams broken, airliners fallen out of the sky, and thousands of automobiles and other consumer products have been recalled - all within recent memory and all the result of design errors. Second, complex hardware designs have correspondingly complex and expensive build phases. As a result, the ability to manufacture such systems limits the number of companies that produce truly complex hardware designs. No such limitations exist for software. There are hundreds of software organizations, and thousands of very complex software systems in existence. Both the number and the complexity are growing daily. This means that the software industry is not likely to find solutions to its problems by trying to emulate hardware developers. If anything, as CAD and CAM systems has helped hardware designers to create more and more complex designs, hardware engineering is becoming more and more like software development.

Designing software is an exercise in managing complexity. The complexity exits within the software design itself, within the software organization of the company, and within the industry as a whole. Software design is very similar to systems design. It can span multiple technologies and often involves multiple sub-disciplines. Software specifications tend to be fluid, and change rapidly and often, usually while the design process is still going on. Software development teams also tend to be fluid, likewise often changing in the middle of the design process. In many ways, software bears more resemblance to complex social or organic systems than to hardware. All of this makes software design a difficult and error prone process. None of this is original thinking, but almost 30 years after the software engineering revolution began, software development is still seen as an undisciplined art compared to other engineering professions.

The general consensus is that when real engineers get through with a design, no matter how complex, they are pretty sure it will work. They are also pretty sure it can be built using accepted construction techniques. In order for this to happen, hardware engineers spend a considerable amount of time validating and refining their designs. Consider a bridge design, for example. Before such a design is actually built the engineers do structural analysis; they build computer models and run simulations; they build scale models and test them in wind tunnels or other ways. In short, the designers do everything they could think of to make sure the design is a good design before it is built. The design of new airliner is even worse; for those, full scale prototypes must be built and test flown to validate the design predictions.

It seems obvious to most people that software designs do not go through the same rigorous engineering as hardware designs. However, if we consider source code as design, we see that software designers actually do a considerable amount of validating and refining their designs. Software designers do not call it engineering, however, we call it testing and debugging. Most people do not consider testing and debugging as real "engineering"; certainly not in the software business. The reason has more to do with the refusal of the software industry to accept code as design than with any real engineering difference. Mock-ups, prototypes, and bread-boards are actually an accepted part of other engineering disciplines. Software designers do not have or use more formal methods of validating their designs because of the simple economics of the software build cycle.

Revelation number two: it is cheaper and simpler to just build the design and test it than to do anything else. We do not care how many builds we do - they cost next to nothing in terms of time and the resources used can be completely reclaimed later if we discard the build. Note that testing is not just concerned with getting the current design correct, it is part of the process of refining the design. Hardware engineers of complex systems often build models (or at least they visually render their designs using computer graphics). This allows them to get a "feel" for the design that is not possible by just reviewing the design itself. Building such a model is both impossible and unnecessary with a software design. We just build the product itself. Even if formal software proofs were as automatic as a

compiler, we would still do build/test cycles as formal proofs have never been of much practical interest to the software industry.

This is the reality of the software development process today. Ever more complex software designs are being created by an ever increasing number of people and organizations. These designs will be coded in some programming language and then validated and refined via the build/test cycle. The process is error prone and not particularly rigorous to begin with. The fact that a great many software developers do not want to believe that this is the way it works compounds the problem enormously.

Most current software development processes try to segregate the different phases of software design into separate pigeon-holes. The top level design must be completed and frozen before any code is written. Testing and debugging are necessary just to weed out the construction mistakes. In between are the programmers, the construction workers of the software industry. Many believe that if we could just get programmers to quit "hacking" and "build" the designs as given to them (and in the process, make fewer errors) then software development might mature into a true engineering discipline. Not likely to happen as long as the process ignores the engineering and economic realities.

For example, no other modern industry would tolerate a rework rate of over 100% in its manufacturing process. A construction worker who cannot build it right the first time, most of the time, is soon out of a job. In software, even the smallest piece of code is likely to be revised or completely rewritten during testing and debugging. We accept this sort of refinement during a creative process like design, not as part of a manufacturing process. No one expects an engineer to create a perfect design the first time. Even if he does, it must still be put through the refinement process just to prove that it was perfect.

If we learn nothing else from Japanese management techniques, we should learn that it is counter-productive to blame the workers for errors in the process. Instead of continuing to force software development to conform to an incorrect process model, we need to revise the process so that it helps rather than hinders efforts to produce better software. This is the litmus test

of "software engineering." Engineering is about how you do the process, not about whether the final design document needs a CAD system to produce it.

The overwhelming problem with software development is that everything is part of the design process. Coding is design, testing and debugging are part of design, and what we typically call software design is still part of design. Software may be cheap to build, but it is incredibly expensive to design. Software is so complex that there are plenty of different design aspects and their resulting design views. The problem is that all the different aspects interrelate (just like they do in hardware engineering). It would be nice if top level designers could ignore the details of module algorithm design. Likewise, it would be nice if programmers did not have to worry about top level design issues when designing the internal algorithms of a module. Unfortunately, the aspects of one design layer intrude into the others. The choice of algorithms for a given module can be as important to the overall success of the software system as any of the higher level design aspects. There is no hierarchy of importance among the different aspects of a software design. An incorrect design at the lowest module level can be as fatal as a mistake at the highest level. A software design must be complete and correct in all its aspects, or all software builds based on the design will be erroneous.

In order to deal with the complexity, software is designed in layers. When a programmer is worrying about the detailed design of one module, there are probably hundreds of other modules and thousands of other details that he cannot possibly worry about at the same time. For example, there are important aspects of software design that do not fall cleanly into the categories of data structures and algorithms. Ideally, programmers should not have to worry about these other aspects of a design when designing code.

This is not how it works, however, and the reasons start to make sense. The software design is not complete until it has been coded and tested. Testing is a fundamental part of the design validation and refinement process. The high level structural design is not a complete software design; it is just a structural framework for the detailed design. We have very limited

capabilities for rigorously validating a high level design. The detailed design will ultimately influence (or should be allowed to influence) the high level design at least as much as other factors. Refining all the aspects of a design is a process that should be happening throughout the design cycle. If any aspect of the design is frozen out of the refinement process, it is hardly surprising that the final design will be poor or even unworkable.

It would be nice if high level software design could be a more rigorous engineering process, but the real world of software systems is not rigorous. Software is too complex and it depends on too many other things. Maybe some hardware does not work quite the way the designers thought it did, or a library routine has an undocumented restriction. These are the kinds of problems that every software project encounters sooner or later. These are the kinds of problems discovered during testing (if we do a good job of testing), for the simple reason that there was no way to discover them earlier. When they are discovered, they force a change in the design. If we are lucky, the design changes are local. More often than not, the changes will ripple through some significant portion of the entire software design (Murphy's Law). When part of the effected design cannot change for some reason, then the other parts of the design will have to be weakened to accommodate. This often results is what managers perceive as "hacking", but it is the reality of software development.

For example, I recently worked on a project where a timing dependency was discovered between the internals of module A and another module B. Unfortunately, the internals of module A were hidden behind an abstraction that did not permit any way to incorporate the invocation of module B in its proper sequence. Naturally, by the time the problem was discovered, it was much too late to try to change the abstraction of A. As expected, what happened was an increasingly complex set of "fixes" applied to the internal design of A. Before we finished installing version 1, there was the general feeling that the design was breaking down. Every new fix was likely to break some older fix. This is a normal software development project. Eventually, we went for a change in the design.

On any software project of typical size, problems like these are guaranteed to come up. Despite all attempts to prevent it, important details will be

overlooked. This is the difference between craft and engineering. Experience can lead us in the right direction. This is craft. Experience will only take us so far into uncharted territory. Then we must take what we started with and make it better through a controlled process of refinement. This is engineering.

As just a small point, all programmers know that writing the software design documents after the code instead of before, produces much more accurate documents. The reason is now obvious. Only the final design, as reflected in code, is the only one refined during the build/test cycle. The probability of the initial design being unchanged during this cycle is inversely related to the number of modules and number of programmers on a project. It rapidly becomes indistinguishable from zero.

In software engineering, we desperately need good design at all levels. In particular, we need good top level design. The better the early design, the easier detailed design will be. Designers should use anything that helps. Structure charts, Booch diagrams, state tables, etc. - if it helps, then use it. We must keep in mind, however, that these tools and notations are not a software design. Eventually, we have to create the real software design, and it will be in some programming language. Therefore, we should not be afraid to code our designs as we derive them. We simply must be willing to refine them as necessary.

There is as yet no design notation equally suited for use in both top level design and detailed design. Ultimately, the design will end up coded in some programming language. This means that top level design notations have to be translated into the target programming language before detailed design can begin. This translation step takes time and introduces errors. Rather than translate from a notation that may not map cleanly into the programming language of choice, programmers often go back to the requirements and redo the top level design, coding it as they go. This, too, is part of the reality of software development.

It is probably better to let the original designers write the original code, rather than have someone else translate a language independent design later. What we need is a unified design notation suitable for all levels of design.

In other words, we need a programming language that is also suitable for capturing high level design concepts. This is where C++ comes in. C++ is a programming language suitable for real world projects that is also a more expressive software design language. C++ allows us to directly express high level information about design components. This makes it easier to produce the design, and easier to refine it later. With its stronger type checking, it also helps the process of detecting design errors. This results in a more robust design, in essence a better engineered design.

Ultimately, a software design must be represented in some programming language, and then validated and refined via a build/test cycle. Any pretense otherwise is just silliness. Consider what software development tools and techniques have gained popularity. Structured programming was considered a breakthrough in its time. Pascal popularized it and in turn became popular. Object oriented design is the new rage and C++ is at the heart of it. Now think about what has not worked. CASE tools? Popular, yes; universal, no. Structure charts? Same thing. Likewise, Booch diagrams, object diagrams, you name it. Each has its strengths, and a single fundamental weakness - it really isn't a software design. In fact the only software design notation that can be called widespread is PDL, and what does that look like.

This says that the collective subconscious of the software industry instinctively knows that improvements in programming techniques and real world programming languages in particular are overwhelmingly more important than anything else in the software business. It also says that programmers are interested in design. When more expressive programming languages become available, software developers will adopt them.

Also consider how the process of software development is changing. Once upon a time we had the waterfall process. Now we talk of spiral development and rapid prototyping. While such techniques are often justified with terms like "risk abatement" and "shortened product delivery times", they are really just excuses to start coding earlier in the life cycle. This is good. This allows the build/test cycle to start validating and refining the design earlier. It also means that it is more likely that the software designers that developed the top level design are still around to do the detailed design.

As noted above - engineering is more about how you do the process than it is about what the final product looks like. We in the software business are close to being engineers, but we need a couple of perceptual changes. Programming and the build/test cycle are central to the process of engineering software. We need to manage them as such. The economics of the build/test cycle, plus the fact that a software system can represent practically anything, makes it very unlikely that we will find any general purpose methods for validating a software design. We can improve this process, but we cannot escape it.

One final point: the goal of any engineering design project is the production of some documentation. Obviously, the actual design documents are the most important, but they are not the only ones that must be produced. Someone is eventually expected to use the software. It is also likely that the system will have to be modified and enhanced at a later time. This means that auxiliary documentation is as important for a software project as it is for a hardware project. Ignoring for now users manuals, installation guides, and other documents not directly associated with the design process, there are still two important needs that must be solved with auxiliary design documents.

The first use of auxiliary documentation is to capture important information from the problem space that did not make it directly into the design. Software design involves inventing software concepts to model concepts in a problem space. This process requires developing an understanding of the problem space concepts. Usually this understanding will include information that does not directly end up being modeled in the software space, but which nevertheless helped the designer determine what the essential concepts were, and how best to model them. This information should be captured somewhere in case the model needs to be changed at a later time.

The second important need for auxiliary documentation is to document those aspects of the design that are difficult to extract directly from the design itself. These can include both high level and low level aspects. Many of these aspects are best depicted graphically. This makes them hard to

include as comments in the source code. This is not an argument for a graphical software design notation instead of a programming language. This is no different from the need for textual descriptions to accompany the graphical design documents of hardware disciplines. Never forget that the source code determines what the actual design really is, not the auxiliary documentation. Ideally, software tools would be available that post processed a source code design and generated the auxiliary documentation. That may be too much to expect. The next best thing might be some tools that let programmers (or technical writers) extract specific information from the source code that can then be documented in some other way. Undoubtedly, keeping such documentation up to date manually is difficult. This is another argument for the need for more expressive programming languages. It is also an argument for keeping such auxiliary documentation to a minimum and keeping it as informal as possible until as late in the project as possible. Again, we could use some better tools; otherwise we end up falling back on pencil, paper, and chalk boards.

To summarize:

- Real software runs on computers. It is a sequence of ones and zeros that is stored on some magnetic media. It is not a program listing in C++ (or any other programming language).
- A program listing is a document that represents a software design. Compilers and linkers actually build software designs.
- Real software is incredibly cheap to build, and getting cheaper all the time as computers get faster.
- Real software is incredibly expensive to design. This is true because software is incredibly complex and because practically all the steps of a software project are part of the design process.
- Programming is a design activity - a good software design process recognizes this and does not hesitate to code when coding makes sense.
- Coding actually makes sense more often than believed. Often the process of rendering the design in code will reveal oversights and the need for additional design effort. The earlier this occurs, the better the design will be.
- Since software is so cheap to build, formal engineering validation methods are not of much use in real world software development. It is

easier and cheaper to just build the design and test it than to try to prove it.

- Testing and debugging are design activities - they are the software equivalent of the design validation and refinement processes of other engineering disciplines. A good software design process recognizes this and does not try to short change the steps.
- There are other design activities - call them top level design, module design, structural design, architectural design, or whatever. A good software design process recognizes this and deliberately includes the steps.
- All design activities interact. A good software design process recognizes this and allows the design to change, sometimes radically, as various design steps reveal the need.
- Many different software design notations are potentially useful - as auxiliary documentation and as tools to help facilitate the design process. They are not a software design.
- Software development is still more a craft than an engineering discipline. This is primarily because of a lack of rigor in the critical processes of validating and improving a design.
- Ultimately, real advances in software development depend upon advances in programming techniques, which in turn mean advances in programming languages. C++ is such an advance. It has exploded in popularity because it is a mainstream programming language that directly supports better software design.
- C++ is a step in the right direction, but still more advances are needed.

1.2. Explain the term 'architecture of the application'?

Architecture is the set of rules (or framework) to bring in some common way of assembling or using various components in the application. This helps in bringing consistency between codes developed by various developers in the team.

1.3. What is design?

Design: a Plan
Design as a noun refers to the product of the design process. The product (plan) is constructed from various design elements.

A plan is a hierarchical description of the components (elements) and their relationships in a proposed product.

Software elements

- Modules, procedures, functions, instruction
- Data types

Software relationships

- Call structure
- Visibility
  - Nesting structure, visibility
  - Inheritance
- Objects - data and functions
- E/R and UML diagrams
- Patterns
- Web site
  - navigational structure
- Web page
  - page layout

1.4. What is software?

Software is:

- an implementation of a mathematical function
- a sequence of state changes (a thread of control)
- a simulation
- an executable theory
- a set of communicating processes
- a set of interacting objects

Software is designed to:

- replace another system
  - through reverse engineering or
  - automation of preexisting system
- simulate
  - an existing system
  - explore alternative systems
- provide a previously unavailable service

In any case, software is analogous to a scientific theory with the added advantage of being executable facilitating its own testing.

1.5. Differentiate between the terms 'Design Patterns', 'Framework' and 'Architecture'.

Design Pattern: The various solutions arrived at for the known problem. This helps to avoid re-inventing the wheel. The risk-free solution can be easily used by others. For example, singleton is the design pattern that you can use instantly to enforce one object per class. You do not need to think of this on your own.
Framework: A framework is a structure or set of rules, used to solve or address complex issues. It is basically a reusable design for the J2EE applications. For example, Struts, JSF, etc., are the frameworks. You can use these frameworks based on the requirements of your application and each has its own set of advantages.

Architecture: It is a design that describes how the various components in the application fit together. For example, MVC is an architecture which is helpful to define how the components interact within the application.

1.6. Define the terms authentication and authorization.

Authentication is the process/rule that validates if the user is allowed to access the system or not.
Authorization is the process/rule that validates if the user is allowed to access the particular part of the system or not. This occurs after user's successful authentication.

1.7. How do you differentiate true designers and architects from other software developers?

This may be the hardest question to answer, and the most necessary. A real designer or architect, someone who does not just hack a bunch of software together, is worth more to your company than you can pay him or her. A real designer or architect can translate the vision of what the product has to be into the here-and-now, no matter what development practices you use. The agile practices help more of us become better designers and architects than the top-down, decomposition practices. Good designers and architects

spend time in their heads. They tend to be more introspective than other people.

1.8. What's the most powerful language you have ever used? Why?

I would listen for LISP or Java. If I hear C or C++, I worry about the exposure this candidate has had to alternative techniques. C and C++ are not particularly powerful. In my opinion, C and C++ can lead to bad designs if the candidate has not been exposed or has not worked with alternative languages.

1.9. Do you typically generate alternative designs?

The best architects and designers generate at least three alternatives.

1.10. How do you choose between these designs?

Listen for at least 2 or 3 stories about how the candidate made decisions, and with whom. Did the candidate make decisions alone? With others? When I have worked with great designers, they chose to bounce ideas off their peers. As a project manager or engineering manager, they would explain the differences to me, but I was not always part of the decision-making process. It depended on the consequences of that decision (who else the decision affected).

1.11. Tell me about the last major design you did. What was the purpose? What happened?

Listen for the candidate's assessment of risks, who the candidate worked with, how the design worked, when the candidate found problems.

1.12. If you had to design for our product (talk about your product here), what approach would you take?

Listen for language choice, type of architecture choice, type of investigation, whether the candidate would prototype, if the candidate would work bottom up or top down. Things to listen for:

1. A lack of introspection. Good designers and architects spend time in their heads. They tend to be more introspective than other people. If a candidate says that he or she has not thought much about how they

design, I wonder about their level of design and architecture expertise.

2. If a designer tells me a first design is the “obvious first and only choice.” I do not buy it. We edit writing, we should edit our designs.
3. A lack of multiple computer language experience. I do not want a designer who does not understand that the technical choices we make about the product radically influence the product architecture and the choice of lifecycle and practices.

Great designers are not born; they are created from experience and learning from experience. Make sure you think about what you need from an architect or designer before you even phone screen a candidate. The value of a great designer or architect is immense - much more than you pay that person.

1.13. What are the productivity measurement considerations?

When we think about manufacturing work, we measure labor productivity as the ratio of the output of goods and services to the labor hours devoted to the production of that output, output per hour.

The project requirements are a tradeoff of how much (feature set), when (time to market) and how good (defect levels). That is the reason we cannot use output per hour as a software (or any knowledge worker) productivity measurement. Here is why: If you do not care how good a deliverable is, I can have it for you almost immediately. It would not work, but my “productivity” if all you consider is when I say the deliverable is complete, is very high. (Not much time spent, so the ratio is high.)

If we want to start measuring developer or tester productivity, we have to define output per hour or whatever time period you desire.

For developers, we can measure designs considered, lines of code generated, the number of defects generated along with the code, unit tests generated, unit tests run, how good that output is, and the time it took the developers to generate all that output.

For testers, we can measure test designs considered, number of tests generated, attempted, and run, test logs, defects detected, defects reported, number of measurements provided back to the developers as information, how good that output is, and all the time it took the testers to generate all that output.

Not so simple, eh? If we just knew how to measure how good our work was, we would have a chance to measure individual productivity. The more I think about productivity, the more I know it is a project-by-project thing, not a person-by-person thing. Yes, some people have more effective output than others. And I am not sure that matters.

1.14. How do you define productivity?

Too many managers asks for help on knowing how "good" their people are. When I ask more questions, such as "What does good mean to you?", they say they want to know who is most productive. Then I walk through this analysis with them:

Productivity is not about the number of lines of code, the number of tests, the number of defects reported, or the number of requirements defined. Productivity is how long it takes the entire team to create a usable product, and for whom that product is usable. It usually takes us a few emails to get past that statement. That is because these managers now realize a single-dimension measurement is inadequate for measuring a person's productivity, and that on a project, it is the productivity of the project, not the productivity of a given person that matters.

For you lifecycle/process aficionados, that is why agile projects, staged-delivery, evolutionary delivery lifecycles, and critical chain project management works. Each of those lifecycles or techniques focuses on getting work out of the project, not waiting until the entire project is complete.

The idea of continual trend measurement is a good one. Taking the few vital measurements daily (or weekly or monthly) in a project, and for some period for people management is a very good idea. Knowing where you start is critical, which is why planning is so useful. Knowing where you end

up is also critical, but measuring in the middle is even more important. Measuring in the middle helps you complete the work to obtain the result you want.

Continuous measurement of vital signs helps you see when things are starting to go awry. If your project has an area where the defects start to increase or the number of reviews starts to decrease, or the estimations are off (either way), you have an opportunity to continue to watch the project or take some action based on the measurements. If you do not measure in the middle, you are surprised by the result.

One of my favorite project measurements is the number of people I need on a project and the number of people I actually have on a project. I find staffing curves help me organize the work breakdown structure (WBS) in different ways, and help me talk about potential project organizations differently with management. Here's a project staffing table:

Month	Planned	Actual
1	2	2
2	4	2
3	6	4
4	6	6
5	6	6
6	6	8
7	not planned	8
8	not planned	8
total person-months	30	48

They'd originally planned a 6-calendar-month 30-person-month project. By the time I arrived (month 6, when they realized they weren't going to make it), the best we could do was an 8-calendar-month, 48-person-month project. During the senior management debrief, a bunch of the senior managers wrung their hands and asked why they couldn't make it. I showed them this table, and asked if they'd checked with the project manager at month 3. At month 3, it was clear the project they had wasn't the same one

they had planned. If they'd measured staffing (as opposed to trying to push to meet milestones), they would have seen this.

Product-projects are not the only ones that require interim measurement. Any cultural change "project" requires interim measurement. In one organization, we changed the culture from a "let's have a meeting but not agendas or action items" to using meetings to come to agreement on decisions and track action/obstacle progress. Here, we measured the number of meetings per week, and the number of action items accomplished per week. As long as the number of action items per week from the meeting continued to go up, it was ok if the number of meetings went up. If the number of meetings went up, but the action items did not, we sent email like this: "Last week the total number of meetings increased. The number of action items did not. Please make sure you track your action items, and if you're having trouble accomplishing your to-dos, do not be afraid to ask for help."

Measure your work in the middle, looking for trends that will help you understand progress and health of your effort. Do not unnecessarily disturb the people, but make sure you have incorporated appropriate measurements.

1.15. What is system architecture?

System architecture is the process of partitioning a software system into smaller parts. The parts can be sub-systems, modules, components, or programs.

1.16. What is most critical activity in software development life cycle?

Architecture and design is the most critical activity in software development life cycle.

1.17. What are the steps in defining architecture?

Starting on a new project is very exciting. You start with a brand new problem and have to define the complete hardware and software architecture for the product. This stage presents its own challenges too. It is a big challenge to convert the system requirements into an architecture that

will handle the requirements. In this article we will divide the process of architecture design into simple steps.

Architecture design can be de divided into the steps given below:

1. Analyze the requirements
2. Define use cases for the system
3. Identify processors/modules to implement the use cases
4. Select operating system and hardware platform
5. Assign requirements to individual processors/modules
6. Define sequence diagrams at processor level

Note that architecture design is an iterative process. When you design your architecture you would move through the steps iteratively. For example, after analyzing the requirements and defining the use cases you might find that there are some aspects of the requirements you did not understand. So you would go back to analyzing the requirements.

Analyze the requirements
Entry Criterion: Customer requirements are available
Exit Criterion: System specifications are reviewed

The first step is to understand the requirements. This appears to be a simple step, you just read the requirements and understand it. But the truth is this could be the most difficult step of the whole process. The reason is that customer requirements may be written in a terse language that may need clarifications from the customer. It is a good idea to document the outcome of this analysis as a specification for the project. Essentially specifications are the customer requirements written in a language your team understands. Often the specifications would have more detail that the original customer requirements. At this stage, getting the system specifications reviewed by the customer will make sure that the customer and you are at the same plane.

Define use cases for the system
Entry Criterion: Reviewed system specifications are available
Exit Criterion: System level use cases are available

Once the system specification is available, develop system level use cases. The use cases at this level, consider the system to be a monolithic block. All interactions between the users of the system and the system itself are clearly documented. This process can itself be divided to into the following steps:

1. Identify the entities that will be interacting with the system
2. Define the interfaces between these entities and the system
3. Identify the most important scenarios for interactions between the users and the system
4. Define the use case interactions between the users and system

An Example

Let's consider the example of a digital switching system (telephone exchange) to better understand the steps involved.

1. Subscribers and other switches are the two entities that will be interacting with a switch.
2. Subscribers will use the system to setup and receive calls using a standard phone line. Other switches will originate and terminate calls on the switch.
3. Important scenarios for the system are:
    - Subscriber originates a call (successful, unsuccessful)
    - Subscriber receives a call (successful, unsuccessful)
    - Other switch originates a call (successful, unsuccessful)
    - Other switch terminates a call (successful, unsuccessful)
4. Define each of the scenarios that was identified in step 3. An example is given below.

Subscriber originates a call (use case)

1. Subscriber lifts the handset
2. System gets an origination request
3. System feeds dial tone
4. Subscriber hears dial tone
5. Subscriber dials the digits
6. Switch removes dial tone
7. Switch routes the call to another switch by sending origination
8. etc.

Identify processor/modules to implement the use cases

Entry Criterion Reviewed system use cases are available
Exit Criterion Hardware processor/modules have been identified and basic functionality has been assigned to the processors.

Once you have use cases for major scenarios, you are ready to move to the next step of identifying modules and processors that would be required to handle the system's functionality. Your selection of the hardware components should depend on the level of experience you have in executing similar projects. Here are a few guidelines:

- If you have an existing system which implements similar functionality, it might be a good idea to reuse the architecture from that project.
- If this is a brand new project, consider if the functionality can be implemented by a single processor. A single processor system would be the easiest to implement and debug. (even if you have to provide a 1-for-1 redundant machine).
- If the nature of the application or the performance requirements necessitate a multiple processor design, select the minimum number of processors that would be required to implement it. (Keep in mind that adding more processors to your system will complicate both hardware and software development).
- In a multiple processor design, separate out the processors that will control and manage the system from the processors that will perform the main operations. It is preferable to have just one processor control and manage the system, thus localizing the complexity of managing the system to a single processor. All other processors in the system handle the real work load of the system.
- Design a multiple processor system in such a way that it can scale from a small system to a large one by adding processors. Thus the customer can start with a small number of processors when the system is initially introduced. The system can then grow as the traffic on the system increases.

Select operating system and hardware platform
Entry Criterion Processor and modules implementing the system have been decided.
Exit Criterion Hardware and software platforms have been selected.

Selection of the operating system and hardware platform is based on the following factors:

- Technical merits (performance, stability) of the hardware/software platform
- Special requirements from the platform (e.g. does the application require a platform that supports a real-time operating system)
- In-house experience with the hardware/software platform
- Financial health of the companies backing the hardware/software platform (you don't want to be stuck with a platform from a company that may not be around to support the product)
- Company's roadmap for the product (you don't want to select a platform which does not have a clearly defined upgrade path)
- Quality of developer tools available for the platform
- Cost of training/availability of developers for the platform
- Head room in the software/hardware platform to grow with your needs (for example, don't pick a hardware platform when you know that the CPU utilization of the processor will be close to 80%, leaving no room for future growth)
- Does the platform require additional in-house development? A completely off-the-shelf platform is going to be a lot cheaper to work with than a in-house designed custom platform.
- Availability of device drivers for the platform (Can you afford to develop customer device drivers for every device that will interface with your platform?)
- Protocol support in the platform (e.g. does the platform support TCP/IP, HTTP, UDP etc.?)

Operating system and hardware platform selections can be very difficult to make. You will find members of your team deeply divided on the selection of the operating system. The best way to introduce some objectivity into decision making might be to consider the factors listed above and assign them weights. Then rate the proposed platform on all these factors.

Assign requirements to individual processors/modules
Entry Criterion Processor and modules implementing the system have been decided and operating system and hardware platform have been selected.

Exit Criterion Complete functionality of the system has been partitioned into different processors. All system specifications have been assigned to processors that constitute the system.

We have identified the modules/processors previously. Once the final hardware and software platform selections have been made, go through the system specifications and assign them to individual modules/processors in the system. A new revision of the system specifications should be produced with these inputs.

Define sequence diagrams at processor level
Entry Criterion System specifications have been partitioned to different processors.
Exit Criterion Processor level sequence diagrams have been produced for all the system use cases.

We consider this to be the final step in architecture design. Here the architects see if the architecture handles all the system level use cases. During this step, system level use cases will give way to processor level sequence diagrams. An example of such a conversion is given below. "Subscriber originates a call" use case is developed into a sequence diagram.

Subscriber originates a call (sequence diagram)

1. Subscriber lifts the handset
2. Originating call processor gets an origination request
3. Originating call processor allocates and assigns a digit collector module.
4. Originating call processor requests dial tone generator to feed dial tone.
5. Dial tone generator feeds dial tone
6. Subscriber hears dial tone
7. Subscriber dials the digits
8. Digit collector collects the first digit
9. Digit collector forwards the first digit to the originating call processor

10. Originating call processor requests the dial tone generator to disconnect the dial tone.
11. All other digits are collected in a similar fashion.
12. Originating call processor obtains routing information from the call routing processor.
13. Originating call processor communicates with the terminating call processor to further route the call.
14. etc.

You can see here that the level of detail has increased considerably over the corresponding system level use case.

1.18. Why do you need architecture?

A successful architecture forms the platform for strategic advantage. By contrast, the lack of architecture bonds the organization inescapably to its past. For most organizations, our legacy is a tortuously tangled slew of haphazard systems born of a time of amazing wizardry but little system discipline. These legacy systems are expensive and hard to change, but replacing them threatens the very "life" of the organization.

Brittle monolithic systems, silo applications, and long and unpredictable development times, are symptomatic of architectural decay which causes huge organizational drag. To break the chains of our corporate legacy and build systems that fit the environment, and adapt with the environment as it changes, we need architecture.

Whether we seek to lead through innovation, customer intimacy or operational excellence, architecture is the essential foundation for agility, responsiveness and effectiveness. Architecture addresses complexity, leaving the team mind-space open to innovation. It is the enabler for reliable systems developed in "Internet time," eCommerce systems that scale, and CRM systems that "Wow" customers with an individualized experience.

Architecture serves both technical and organizational purposes. On the organizational side, the architecture helps in:

- Communicating the high-level design: A number of stakeholders need to understand the system at a fairly gross level. These include higher-level managers, many of the cross-functional team (e.g., marketing, quality assurance, and learning products or user documentation), and may include customers too. Modeling the system at a high level facilitates communication of the high-level system design or architecture. The reduction in detail makes it easier to grasp the assignment of significant system responsibilities to high-level structures. Moreover, it satisfies the constraint that, though seemingly trivial, has important implications for communication—with suitable leveling and nesting, even large and complex architectures can be conveyed using presentation slides and documented using traditional 8.5 x11" paper!
- Providing the system context: The developers (and future maintainers) also need to understand the overall system. In large systems, developers cannot efficiently understand the details of the entire system. They need a detailed understanding of the more narrowly-scoped portions of the system that they work on. But without an understanding of the responsibilities and interdependencies of the higher-level structures, individual development and optimization of the substructures will tend to result in a sub-optimal system. This is both from the point of view of system characteristics like performance, as well as effort in integration and maintenance.
- Work allocation: Where architectures decompose the system into substructures that are relatively independent, have clear responsibilities, and communicate with each other through a limited number of well-defined interfaces, the development work can be partitioned effectively. This allows parallel development work to proceed in relative independence between integration points. This is especially important in large projects, or projects where the teams are geographically dispersed or subcontractors are used. Moreover, since these units tend to be centers of specialization of function or service, they also afford opportunities for skill specialization among developers. This independence and focus makes development more efficient. The design of the system architecture can be viewed as the dual of designing the organization architecture. If this duality is

ignored and the organization architecture is not compatible with the system architecture, then it can influence and degrade the system architecture. This is known as "Conway's Law" (Conway, 1968).

On the technical side, architecture allows us to design better systems:

- Meet system requirements and objectives: Both functional and non-functional requirements can be prioritized as “must have” vs. “high want” vs. “want”, where “must have” identifies properties that the system must have in order to be acceptable. An architecture allows us to evaluate and make tradeoffs among requirements of differing priority. Though system qualities (also known as non-functional requirements) can be compromised later in the development process, many will not be met if not explicitly taken into account at the architectural level.
- Enable flexible distribution/partitioning of the system: A good architecture enables flexible distribution of the system by allowing the system and its constituent applications to be partitioned among processors in many different ways without having to redesign the distributable component parts. This requires careful attention to the distribution potential of components early in the architectural design process.
- Reduce cost of maintenance and evolution: Architecture can help minimize the costs of maintaining and evolving a given system over its entire lifetime by anticipating the main kinds of changes that will occur in the system, ensuring that the system's overall design will facilitate such changes, and localizing as far as possible the effects of such changes on design documents, code, and other system work products. This can be achieved by the minimization and control of subsystem interdependencies.
- Increase reuse and integrate with legacy and third party software: Architecture may be designed to enable and facilitate the (re)use of certain existing components, frameworks, class libraries, legacy or third-party applications, etc.

1.19. Why do you need a layered architecture? Why are layered architectures so useful?

Well, first and most of all they allow you to distinguish and distribute the responsibilities that your application, your code has to deliver value to the end user. That is, the user interface does not contain components or elements that handle business logic. Business logic resides in a separate layer. A data access layer does not have anything to do with presenting the data, it deals with the database.

And if your thinking this is trivial, please note that there are hordes of developers who still write database connection strings and queries in each web page they develop and who still validate the same business rules in each web page they data is presented. You're not alone.

Distinguishing and distributing responsibilities - if done correctly and pragmatically - will deliver a number of benefits to your projects:

- Easier to understand. Code will be easier to understand, not only for the developers who wrote it, but also for the poor bastards who have to maintain the code once delivered. Code is just easier to maintain when you know where to find it. Code becomes traceable.
- Easier to write. Have developed applications for over twenty years, I think I'm entitled to say that a day's work is more fun writing software under architecture. It's cleaner, more serene.
- Easier to test. Although I personally consider testing software not the most fun, I do think that software that is written under architecture is far more modular. This makes it possible to define independent unit much better, which makes unit testing easier and better. In fact, testability is one of the main motives to define your layered architecture. Design it with testing in mind.
- Easier to extend. Last but not least, having a layered architecture in place will allow you to add new features, or change the current features more easily. Adding a new use case to the system, or extend the business rules on a particular domain object much harder if the process or business logic is spread throughout the code.

Layered architecture is great but can also lead to a lot of confusion when people use it too literally. In a lot of cases people think layered architecture means UI on top of Business Logic on top of Data Access on top of the database with all the layers only talking to the layer below. This leads to an

architecture where business objects are not persistence ignorant and leads to very inflexible applications. A better way of looking at things is layers around a center of business or domain objects. You still work with layers but the layers are arranged differently.

1.20. How do you document or present architecture?

There is no standard or one way of presenting a proposed architecture. You may use:

- UML (unified modeling language)
- Graphical representation
- Block diagrams
- Flow charts
- Combination of the above.

1.21. How do you sell architecture to senior management?

You are convinced that architecture is important, but how do you convince senior managers? One approach that is very successful is to integrate their business goals into an architectural vision and present that vision in a way that is compelling to them.

1.22. How do you create an architectural vision?

A compelling architectural vision is the key to the success of an architecting effort. As part of the start-up phase to the architecting project - we call it Init/Commit - the vision serves to:

- build management commitment
- align the architecting team

As the project progresses, the vision guides the architects in their scoping and structuring decisions, and is also used to help build broader support for the architecture in the organization. In creating a compelling vision, it is helpful to understand:

- where we have come from (Graphical History)
- our current context (Context Map)
- our expected future context (Technology Roadmap)
- our desired future state (Desired State Interviews)

Here, we discuss Desired State Interviews as a means to gathering information from key influencers in the organization.

Desired state interviews stimulate creativity and free the person being interviewed from the shackles of present circumstance by placing her “in the future.” From that vantage point, the interviewee is asked to look at the results achieved with the architecture, and the path that was taken to get to there.

Who to Interview: The goal is to create a vision that inspires and motivates. To do that, you need to understand the values of those who you want to champion the vision, and those who you want to buy into the vision. You also want to understand what will be important to your customers in the future. In selecting who to interview, you are trading off the time your team will need to invest in conducting the interviews and analyzing the interview data, with the benefit of making stakeholders feel part of the process and having their inputs play a role in shaping the vision. You should at least include managers, who will be called upon to sponsor the architecture, and influential project managers and engineers. To gain an understanding of where your markets are headed, interview lead customers and members of your future product marketing group.

It is sometimes a challenge when starting out with this technique, to ask colleagues, especially senior managers, to imagine themselves in the future. Put aside your preconceptions- this is not child’s play. Rest assured, in our experience senior management tends to be most comfortable with this very rewarding technique. Peer engineers may be less receptive, but you can practice with fellow team-mates until you are good at helping a person be comfortable in the imagined future state.

Interview Guidelines: Pick a time in the future when the architecture is complete and its benefits are being reaped. Ask your interviewee to imagine herself at that future time, and ask her to talk about the results she sees and what she hears others saying about the architecture. Help her to “stay” in the future, by staying in the future yourself as you ask her questions to stimulate her imagination or to get clarifications of what she is reporting. Ask her to share details. Encourage her to use all her senses, and let

imagery and ideas flow unchecked. The more richly imagined, the more inspiring this input will be to the architectural vision.

Once you are sure that the interviewee is comfortable imagining that future state, ask the following questions:

- What do you have, now that you have the architecture in place?
- How do you know you have it?
- What does this get you?

The first of these questions identifies results, the second evidence, the third the underlying value to the interviewee. This latter is the most valuable information of all, as it will help you create a vision that truly motivates.

Tips:

- Check to make sure the interviewee is imagining the vision already accomplished, and that you are treating it as such. You should be using the present tense to talk about that future state, and the past tense to talk about what was done to get there.
- Avoid evaluations!
- Encourage people to "just make it up" if they seem to have trouble imagining the future state.

Data Analysis: Once your interviewing is done, the team should meet to put together the information you have gathered and

- sort for common vision elements, keeping as much detail as possible
- look for central themes, unifying imagery and metaphors, etc.

1.23. Why do you need enterprise architect?

First, the summary of architect responsibilities:

- Enterprise architects: set the overall vision and framework for the IT environment from a strategic business perspective.
- Solutions architects: design the solution to take advantage of the existing assets, integrate them into the existing environment, follow the enterprise architecture, and solve the business problems of the business owner or unit.
- Infrastructure architects: responsible for creating an architecture that meets the business and service level agreement requirements of the

business owners and supports the applications and solutions that are required to operate their day-to-day businesses.

An enterprise architect (EA) takes a company's business strategy and defines IT systems architecture to support that strategy. To do so, EAs must understand a company's business and be able to dive deeply into technology issues. In recent years, the role has moved out of the banking industry to pop up all over the corporate universe as companies move to align business goals and the IT infrastructure that supports the business and helps achieve those goals.

Why you need one: With more than 50 percent of IT projects typically not achieving their stated goals, having someone to ensure a company's technology objectives are aligned to its business goals is vital. The EA role becomes more important as companies adopt service-oriented architecture (SOA) approaches toward application development. To realize significant cost savings with SOA, issues of software quality and reusability are the key. An EA must be able to see whether the application has been built with quality and with reuse in mind. They don't need to know how to program, but they need to be able to recognize patterns.

The bottom line is that company hires expensive and canny software architects before start of the project. They talk with the business, come up with solutions and make bright key decisions about architecture. But they don’t program. During the course of the project they ensure that developers don’t spoil this great architecture. Now business owners can hire not so bright programmers, who should just follow directives and implement this great architecture. There is even correspondence with theories of some management consultants that programmers are just house painters.

What to look for: Communication is a key skill; self-confidence is a must. Enterprise architects have to talk to both technical developers and business managers. They need to be able to stand up in a meeting and tell the most senior person in the room unwelcome news, like an IT project won't make its deadline. EAs also need to demonstrate they're on the cutting edge of enterprise software and SOA.

Elimination round: Be leery of candidates who claim their IT projects have been 100 percent successful. Look instead for interviewees who willingly discuss projects where they didn't achieve the goal and tell you what they learned from the experience.

1.24. What may be the reasons why you don’t need software architect?

The Architect (dedicated non-programming technical decision maker and problem solver for business):

1. Has outdated programming knowledge and experience, loss of touch with modern development approaches and practices.
2. Don’t program and don’t know much about evolving system internals, but makes key technical decisions. Often has completely irrelevant and unreal picture what is happening with the system.
3. Tends to complex, premature and generic solutions when the system is still in infancy and nothing is clear. Applies latest modern buzzword technologies as SOA, MDA, SaaS, Software Factories, etc. which look so beautiful in technical magazines, conferences and CV, but cause unnecessary headache for developers.
4. Plays role of the middleman introducing complexity in coordination and project responsibilities. Represents software team in interactions with business customers reducing communication value for the rest of the team and impacting idea flow.
5. Reduces quality of decisions, which become limited to one perspective; decision making starts lacking diversity, independence and decentralization, which are essential attributes of collective intelligence.
6. Creates tension with developers who experience mismatch between grand design and reality. Often continues pushing design decisions until the system becomes overly complex, difficult to change and becomes completely unusable.
7. Secures job and justifies high salary – becomes authoritative center for solving business problems without much input from the team.
8. Causes loss of sense of ownership, motivation and accountability in developers by detaching them from the key architecture decisions.
9. Concentrates project knowledge and the big picture in one head, limiting (and sometimes preventing) complete understanding for

others.

10. Contributes to creation of specialized IT verticals that hurt relations with the business.

I am sure that some architects bring strong technical leadership, excellent solutions and overall project success. However, the architect role itself, as described by Microsoft and employed by other companies doesn't encourage productive development and effective solutions.

1.25. What is effective architecture? Who is effective architect?

Emerging and evolutionary architecture is a core for a successful agile (and not only) software system, emerging from the implementation of business needs, learning from working code, problems and interactions with people, other systems and operational environment.

Effective architecture:

- Provides stable foundation and integrity for the growing software system, enabling desired system qualities for the business solutions.
- Ensures seamless and consistent integration, well defined interfaces and interaction between subsystems, external systems and operational environment.
- Supports emerging abstractions, key system elements and frameworks for conceptual integrity, efficiency and reuse.

Effective architect:

1. Guards a system against failure, sensing worrying changes in the project dynamic, system code and business requests.
2. Keeps the trinity of system qualities (known from time of Roman empire) in a balance and prevent degradation and entropy:
    - Firmitas (Strength) - the system is reliable, secure, has good performance and easy to support
    - Utilitas (Utility) - the system is usable, meets business needs and add business value every day instead of drifting into technology trenches
    - Venustas (Beauty) - code and system structure are clean, easy to understand, minimal

3. Encourages constant improving of the code design, enhancing system abstractions and structure, removing duplication, defining boundaries and interfaces of the subsystems.
4. Keeps solutions as simple as possible, maintains intellectual control over system and avoids over-engineering.
5. Most important - grows and coaches other developers to become architects

If your company can afford dedicated non-programming technical person, hire an expert for complex technical areas, a coach to share best practices and experience or a technical supporter to coordinate with different external groups and help to resolve problems and make them all part of the team.

But don't take technical decisions and concern about architecture from the development team.

The ultimate solution - every developer covers these architect responsibilities. Hire few experienced, motivated and capable developers and start project with them. Developers will constantly evolve the system based on real needs and shape effective architecture. You can trust that system will be healthy, meet business needs and bring success. Often only few important architecture decisions should be made before start of the project - trust them to the developers, who will implement the system.

1.26. Who, when and how should architecture decisions be made?

So, here is the answer regarding who, when and how should make architecture decisions:

- Who - every developer timely makes architecture decisions and constantly improves evolving architecture
- When - mostly after start of implementation, when system becomes more complex and need design improvements to maintain system qualities, intellectual control and conceptual integrity.
- How - preventing degradation and entropy by making design changes and refactoring; learning from the project dynamic, working code and business requests and adjusting a way how the development team works.

1.27. What is your definition of a well-designed product?

A high-quality design has several general characteristics. If we could achieve all those goals, our design would be considered very well indeed. Some goals contradict other goals, but that is the challenge of design - creating a good set of trade-offs from competing objectives. A good design is one that meets the following criteria:

1. Correctness: Will the design work as intended?
2. Completeness: Is the design adequate for all its intended purpose? Does it meet or exceed the requirements, goals and objectives of the customers?
3. Understandability: Can the design be understood easily by others?
4. Good software engineering practices: Does it follow the well-established good software engineering practices?
5. Ship on time: Can the product be shipped on time with this design?
6. High quality: Would the design result in high quality product?

Some of the salient characteristics of a well-designed product are: low complexity, ease of maintenance, extensibility, reusability, ease of use, portability, performance, scalability, availability, reliability, security, adherence to standards, accessibility, globalization, ability to handle load, and stress, etc.

1.28. What is “good design”?

“Design” could refer to many things, but often refers to “functional design” or “internal design”.

Good internal design is indicated by software code whose overall structure is clear, understandable, easily modifiable, and maintainable; is robust with sufficient error-handling and status logging capability; and works correctly when implemented.

Good functional design is indicated by an application whose functionality can be traced back to customer and end-user requirements. For programs that have a user interface, it is often a good idea to assume that the end user will have little computer knowledge and may not read a user manual or even the on-line help; some common rules-of-thumb include:

- the program should act in a way that least surprises the user
- it should always be evident to the user what can be done next and how to exit
- the program should not let the users do something stupid without warning them.

1.29. What should a software designer consider in the design and implementation of code?

The following are some of the considerations:

- Throughput
- Latency
- Reliability
- Usability
- Resource limitations
- Supportability.

1.30. How do you go about obtaining requirements from your customer?

Business requirements analysis is a high-level assessment for the entire project that addresses the what, where, when, why, who, and how for the project. I would like to obtain requirements using the following structure to the extent possible by discussing with stake holders, key managers and end users:

- Introduction (with Purpose and scope)
- User Requirements and Statement of Problem, that is, what is needed
- Application context
- Assumptions
- Other requirements: Performance (both online and batch), Scalability, Reliability, Audit and Control features, Security considerations, Backup and Recovery, Availability, Fault Tolerance, Usability, Conformance to standards, Hardware and Software requirements, Integration, Geographic location, Reports,
- Acceptance criteria: how the product will be accepted
- Support Considerations: Training, maintenance, documentation
- Time frame or schedules

1.31. How do you make certain you understand what they want?

I would like to verify/validate my understanding of the requirements by developing requirements/functional specification documents as well as developing Object Models, Data Models, Use Cases and Business Scenarios. Once documented, we can use these to review with customers for verification and validation.

1.32. Once you have the requirements, what do you do with them?

I would like to document and develop models so as to validate the requirements with customer and for architect/designer-developer. The next step after verification/validation is the detailed analysis of the requirements. The requirements statement may be incomplete or informal; analysis makes it more precise and exposes ambiguities and inconsistencies. I would develop class diagrams, application partitioning, collaboration graphs, data flow diagrams, data models using techniques like UML, ERD, DFD.

1.33. To what level do you document them? Please provide samples of requirements gathering documentation work that you have done.

I would like to document requirements using Object Models, Data Models, Data Flow Diagrams, Use Cases, Business Scenarios as well as text documents.

1.34. What is “good code”? Or what do you consider to be "good code"?

“Good code” is code that works, is bug free, is readable, understandable, and maintainable. We can measure good code by number of bugs that it contains and how many times it has been sent back to the development team. Some organizations have coding “standards” that all developers are supposed to adhere to, but everyone has different ideas about what is best, or what is too many or too few rules. There are also various theories and metrics, such as McCabe Complexity metrics. It should be kept in mind that excessive use of standards and rules can stifle productivity and creativity. Peer reviews, buddy checks, code analysis tools, etc. can be used to check for problems and enforce standards.

Here are some typical ideas to consider in setting rules or standards; these may or may not apply to a particular situation:

- minimize or eliminate use of global variables.
- use descriptive function and method names: use both upper and lower case, avoid abbreviations, use as many characters as necessary to be adequately descriptive (use of more than 20 characters is not out of line); be consistent in naming conventions.
- use descriptive variable names: use both upper and lower case, avoid abbreviations, use as many characters as necessary to be adequately descriptive (use of more than 20 characters is not out of line); be consistent in naming conventions.
- function and method sizes should be minimized; less than 100 lines of code is good, less than 50 lines is preferable.
- function descriptions should be clearly spelled out in comments preceding a function's code.
- organize code for readability.
- use whitespace generously: vertically and horizontally
- each line of code should contain 70 characters max.
- one code statement per line.
- coding style should be consistent throughout a program (e.g., use of brackets, indentations, naming conventions, etc.)
- in adding comments, err on the side of too many rather than too few comments; a common rule of thumb is that there should be at least as many lines of comments (including header blocks) as lines of code.
- no matter how small, an application should include documentation of the overall program function and flow (even a few paragraphs is better than nothing); or if possible a separate flow chart and detailed program documentation.
- make extensive use of error handling procedures and status and error logging.
- for C++, to minimize complexity and increase maintainability, avoid too many levels of inheritance in class hierarchies (relative to the size and complexity of the application). Minimize use of multiple inheritance, and minimize use of operator overloading (note that the Java programming language eliminates multiple inheritance and operator overloading.)

- for C++, keep class methods small, less than 50 lines of code per method is preferable.
- for C++, make liberal use of exception handlers.

1.35. What are the criteria for components?

The following are some of the considerations:

- May be used by other software elements (clients).
- May be used by clients without the intervention of the component's developers.
- Includes a specification of all dependencies (hardware and software platform, versions, other components).
- Includes a precise specification of the functionalities it offers.
- Is usable on the sole basis of that specification.
- Is composable with other components.
- Can be integrated into a system quickly and smoothly.

1.36. What are design patterns?

A design pattern is a description of communicating objects and classes that are customized to solve a general software design problem within a particular context. A recent survey stated that 52% of embedded projects are late by 4-5 months. Design patterns are generalized solutions to recurring problems and, more importantly, they are a way of capturing and reusing design solutions that have been shown to be effective in different circumstances. Design patterns optimize a system or part of a system and may be characterized in terms of both benefits (aspects optimized) and costs (aspects deoptimized). Most of the work in design patterns has been focused on object-oriented languages, but the techniques and benefits are just as applicable to systems developed in the C language.

1.37. In what ways do design patterns help build better software?

Design patterns help software developers to reuse successful designs and architectures. It helps them to choose design alternatives that make a system reusable and avoid alternatives that compromise reusability through proven techniques as design patterns.

1.38. Why use patterns?

Patterns are used to describe the important ideas in the system that appears in multiple places. Patterns help you to explain why your design is the way it is. It is also useful to describe designs you have rejected and why you rejected them.

1.39. What are the differences between analysis patterns and design patterns?

Analysis pattern are for domain architecture, and design pattern are for implementation mechanism for some aspect of the domain architecture. In brief, analysis patterns are more high level and more (end-user) functional oriented.

1.40. What are singleton patterns?

In software engineering, the singleton pattern is a design pattern that is used to restrict instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system. The concept is sometimes generalized to systems that operate more efficiently when only one object exists, or that restrict the instantiation to a certain number of objects (say, five). Some consider it an anti-pattern, judging that it is overused, introduces unnecessary limitations in situations where a sole instance of a class is not actually required, and introduces global state into an application.

Common uses of singleton patterns:

- The Abstract Factory, Builder, and Prototype patterns can use Singletons in their implementation.
- Facade objects are often Singletons because only one Facade object is required.
- State objects are often Singletons.
- Singletons are often preferred to global variables because:
  - They do not pollute the global namespace (or, in languages with namespaces, their containing namespace) with unnecessary variables.
  - They permit lazy allocation and initialization, where global variables in many languages will always consume resources.

1.41. What patterns are particularly useful in building networked applications?

The best pattern for implementing the network solutions are:

- Proxy Pattern

- Facade pattern
- Observer pattern

1.42. What is event driven programming?

Event-driven programming is a program-design technique that object oriented programmers use for a wide variety of tasks, from GUI programming to XML parsing.

In traditional programming languages, the application, rather than a specific event, controls the portion of the computer code which is executed. In addition, the traditional programming coding executes from the beginning of the code and follows a predefined pathway through the application, calling procedures and functions as needed. Flow charts are often used by programmers to structure the programs. Some examples of traditional programming languages are Pascal, C, FORTRAN, COBOL, and Lisp.

Computer languages such as Visual Basic are known as event-driven programming languages. This means that the user executes an event procedure such as clicking a button to run part of the code. Event-driven programming does not have a predefined pathway in the execution of the code, as opposed to traditional programming. User actions determine the sequence of code executions.

An advantage of this type of programming is that no prestructure for the code is necessary. Its high flexibility allows the user to modify parts of the code easily without interfering with the rest of the code.

1.43. What is reengineering?

Reengineering refers to the process of rethinking the way a business operates in the hope of improving its efficiency not just a little bit, but by orders of magnitude. In essence, it means reinventing the business, discarding the preconceived notions that subtly limit the effectiveness of the business.

1.44. What are factory classes?

Factory classes provide an interface for creating families of related objects. Factory classes are useful when the decision of which class to use must be done at run time and cannot be hard coded during development. Factory classes encapsulate the logic needed to decide which subclass to instantiate and so removes this decision from the application, delegating it to the factory.

A class that implements the “factory-pattern” is used to abstract the creating or locating an object or class from the application. The object factory typically contains methods that return objects implementing a known interface. The actual object type returned need not be known at compile time.

An example of an object factory method is Calendar::getInstance() which returns an instance of the class GregorianCalendar initialized to the current date and localized according to the current settings.

1.45. What is refactoring?

Refactoring is a process articulated and popularized by Martin Fowler for the disciplined restructuring of code. The basic point is to restructure the code in relatively simple, straightforward steps such that the resulting code does precisely the same things but is “better” in various ways such as efficiency, simplicity, understandability, manageability, etc.

1.46. What is factoring?

One of the most important considerations in establishing an initial design framework is to properly factor the design into practical and reusable components that take into account the rules of relational database design. For example, a user address could be represented as a series of individual fields, as an Address class, or as a more generic Location class. A group of classes that all send email could each implement email code, call a static SendMail method in a utility class, or utilize an application-standard Message object. Generally, just as you want to normalize databases, you want to factor out object models as much as possible. However, just like databases, this is not always practical from performance or development overhead considerations, and so compromises are made. Factoring is an

ongoing process and the application should be re-examined periodically to evaluate whether current strategies should be changed.

1.47. What is the difference between an API and a framework?

A framework is a set of classes which handles the flow necessary to perform a complex task, but which requires plug-in classes specific to your application (database, application server), and which allows for additional plug-in classes to extend the functionality. For example, you could have a framework which handles the general flow of an online store. This framework would handle tasks such as "put item in a shopping cart", "generate order on checkout", etc. On the other hand, you would need to provide the classes which enable the framework to persist the shopping cart to the database of your choice, to connect to a credit-card processor, to send orders to a warehouse, among other things.

API (application programming interface) is really no more than a set of method signatures and other information necessary to use a class or set of classes. API is a set of routines available in an application for use by software programmers when designing an application interface. API is totally distinct from the implementation and API by itself has no implementation. For example, a number of vendors have implemented the servlet API, each in a different way.

A class library would have an API, as would a framework. Windows SDK, iPhone SDK and Java JDK are APIs while .Net and J2EE are framework.

1.48. What is SDK?

A Software Development Kit (SDK) includes tools, headers, libraries, sample code, and documentation to help you develop and write to say, Microsoft or J2EE technologies and products. They actually expose much of their code in the form of samples that will allow you to write against their providers.

1.49. What are callback functions?

Callback functions are very common in many APIs. Suppose I have a program that is waiting for an event of some sort. For the purpose of this

example, let us pretend that my program wants a user to press a key on the keyboard. Every time they press a key, I want to call a function which then will determine what to do. The function I am utilizing is a callback function. Every time the user presses a key, my program will call the callback function. A callback function is a function that is called through a function pointer. If you pass the pointer (address) of a function as an argument to another, when that pointer is used to call the function it points to it is said that a call back is made.

1.50. Why should you use callback functions?

Because they uncouple the caller from the callee. The caller does not care who the callee is; all it knows is that there is a callee with a certain prototype and probably some restriction (for instance, the returned value can be int, but certain values have certain meanings).

If you are wondering how is that useful in practice, imagine that you want to write a library that provides implementation for sorting algorithms, such as bubble sort, shell short, quick sort, and others. The catch is that you do not want to embed the sorting logic into your functions, making your library more general to use. You want the client to be responsible to that kind of logic. Or, you want it to be used for various data types (ints, floats, strings, and so on). So, how do you do it? You use function pointers and make callbacks.

A callback can be used for notifications. For instance, you need to set a timer in your application. Each time the timer expires, your application must be notified. But, the implementer of the timer's mechanism does not know anything about your application. It only wants a pointer to a function with a given prototype, and in using that pointer it makes a callback, notifying your application about the event that has occurred. Indeed, the SetTimer() WinAPI uses a callback function to notify that the timer has expired (and, in case there is no callback function provided, it posts a message to the application's queue). SetTimer function creates a timer that executes a function at the specified time-out value.

Another example from WinAPI functions that use callback mechanism is EnumWindow(), which enumerates all the top-level windows on the screen.

EnumWindow() iterates over the top-level windows, calling an application-provided function for each window, passing the handler of the window. If the callee returns a value, the iteration continues; otherwise, it stops. EnumWindows() just does not care where the callee is and what it does with the handler it passes over. It is only interested in the return value, because based on that it continues its execution or not.

Callback functions are inherited from C. Thus, in C++, they should only be used for interfacing C code and existing callback interfaces. Except for these situations, you should use virtual methods or functions, not callback functions.

1.51. Describe Microsoft .NET.

Microsoft .NET is a set of Microsoft software technologies for connecting information, people, systems, and devices. It enables a high level of software integration through the use of Web services - small, discrete, building-block applications that connect to each other as well as to other, larger applications over the Internet.

1.52. What are the characteristics of the two-tier application model?

The database and the server limit performance and scalability. In a two-tier application model, program-to-database communications are used between tiers.

1.53. Describe 3-tier architecture in enterprise application development.

In a 3-tier architecture, an application is broken up into three separate logical layers, each with a well-defined set of interfaces. The presentation layer typically consists of a graphical user interfaces. The business layer consists of the application or business logic, and the data layer contains the data that is needed for the application.

1.54. Describe a reason why one would want to build an application using 3-tier scheme.

Advantages of a 3-tier architecture are:

- Greater scalability and improved performance.
- Flexibility to partition and distribute the application.

1.55. Describe a reason why one would not want to build an application using 3-tier scheme.

Disadvantages of a 3-tier architecture are:

- Management of the application is more difficult because of the distributed pieces.
- Developing the infrastructure can be complex and a major part of the development effort.

1.56. What is middleware?

Middleware is software that resides between two or more types of software and translates information between them. Middleware can cover a broad spectrum of software and generally resides between an application and an operating system, a network operating system, or a database management system. Examples of middleware are CORBA (Common Object Request Broker Architecture) and other object broker programs, network control programs, and computer telephony integration (CTI) servers.

1.57. What is MVC?

A key aspect of the overall application design is how to divide the responsibility of formatting or presenting data from the responsibility of managing and distributing data. The standard mechanism for this is often called the Model-View-Controller (MVC) architecture.

Under MVC, the Model is the data itself - in this case the database, or Entity Beans (which are a representation of the database' data). The Controller is the application server components - that is, classes. The View is the code that actually renders the data into HTML, application UIs, etc. The View code only talks to the Controller code, and the Controller code is responsible for all interactions with the Model. This allows different Views to be created with the same Model or Controller.

This means drawing definite lines between any component concerned with presenting data from components that simply provide data or non-visual

functions. With WebLogic, HTML is generated either via Java Server Pages (JSP) or servlet code. We take the approach of designing the UI by using the APIs as though nothing else existed and writing completely separate components and classes to generate and control the interface of the application.

1.58. What is CORBA?

Common Object Request Broker Architecture. It is designed to make sure that a client's request for a service in a Distributed Object Management System is sent to the proper server.

1.59. What is computer telephony integration (CTI)?

CTI is the means by which a computer system can interact with the telephone system. Telephone activity, such as making a call or answering a call, can be initiated within the enterprise application. Information about calls is passed to the host software like Oracle CRM, which can use this information to respond in a variety of ways.

1.60. What is a modal dialog?

When a modal dialog box becomes visible, the user must deal with it immediately. Values must be entered. Then either the OK or Cancel button must be pressed before any other window can gain focus. Consequently, a modal dialog box exists only for a short time.

1.61. What is a modeless dialog?

A modeless dialog box is a dialog that can stay visible over an extended time. While the modeless dialog is visible, the main window can and usually will regain focus. The main window can receive messages and do considerable processing while the modeless dialog box remains visible.

A very useful type of modeless dialog is one that consists of a list box. Important events that occur in the main window can then be appended to the list box, showing a "history" of what has occurred in the main window. A modeless dialog box gives the programmer another surface on which to provide information to the user. Help is usually implemented as modeless dialog.

1.62. What is MDI?

MDI stands for multiple document interface. The purpose of the MDI frame is to encapsulate all child windows of an application. Examples - File Manager, Excel, MS PowerPoint. Characteristics of MDI:

- Non-modal - Window does not have to be closed before opening up another window
- Sheets in a frame
- Work gets done via menus
- Toolbars (optional)

1.63. What is SDI?

SDI stands for single document interface. Examples - Notepad, Calculator, PC Paintbrush. Characteristics of SDI:

- Modal - Window must be closed before opening up another window
- Work gets done via menus and command buttons

1.64. Which GUI style is best?

SDI

- Computer neophytes
- Repetitive tasks
- Linear work flow

MDI

- Computer-literate information workers
- Concurrent access to multiple tasks
- Non-linear work flow

1.65. What are the advantages of popup menus?

Advantages of popup menus include:

- Accessed by right mouse click
- Instant access to common actions and attributes
- Suitable for advanced Windows users

1.66. What does effective GUI design requires?

Effective GUI design requires:

- Understanding of GUI principles
- User analysis and interface design skills
- Close interaction with the actual end-user
- GUI standards and guidelines

1.67. What are GUI design objectives?

To routinely make users more effective in accomplishing their goals and tasks by:

- Incorporating user analysis into methodology
- Tailoring GUI design based on user analysis
- Knowing where to expand effort in various aspects of user interface development.

GUI design principles:

- Different views:
  - User - Conceptual Model - How the user “thinks” it works.
  - Developer - Implementation Model - How it “really” works.
- Leverage what your users already know:
  Objectives:
  - Minimize learning curve, training costs, support calls
  - Provide consistency
- Methods:
  - Set GUI standards based on common user interfaces
  - Employ familiar concepts (invoices, work orders, etc.)
  - Beware of using an entirely new metaphor (form over functionality diminishes value)
  - Focus on user’s goal (going home on time, getting work done easily, etc.)
- Usability Testing:
  - Involve the actual users
- Determining user goals in three steps:
  1. Determine the various user roles (listing them helps you identify the types of people who will use your software)
  2. Ask questions
  - What are the goals of a person in this role?
  - What does success for a person in this role mean?
  3. Verify/document each user’s goals via interview with them

1.68. Why GUIs fail?

Programmers and designers:

- Design for themselves, not for users
- Want the application to control the user
- Think users read documentation
- Are unaware of the consistency issues and standards

1.69. What are the key GUI strategies?

Three key GUI strategies are:

1. Leverage what users already know
2. Modify the development life cycle
3. Develop a corporate GUI awareness

1.70. What is the procedure for application migration?

The procedure for the migration of large and mission-critical corporate applications consists of the following steps:

1. Business Analysis and Migration Planning: Establish and analyze business and technical requirements that will drive and guide the migration projects.
2. Develop and Justify a Migration Architecture: Establish a migration architecture that selects the best migration approach.
3. Implementation: Acquire or build the software needed to enable migration.
4. Deployment and Support Considerations: Migrate the selected application components to the target application.

1.71. What are DLLs? What are their advantages?

Dynamic link libraries (DLLs) are a key feature of Microsoft Windows. They are libraries of executable programs, with well-defined interfaces. As their name suggests, applications link with them dynamically at run time rather than statically at compile time. There are many advantages to this architecture:

- When multiple applications use a DLL simultaneously, they share code, which means there is only one copy of the DLL code in memory.

- The application is separate from the DLL, so you can change the DLL without changing the application or vice versa.
- All DLLs that provide a compatible API also provide a compatible application binary interface (ABI).

The last advantage on the list is the most significant, because an ABI allows portability of executable, not just source code. This means that once you have compiled and linked source code to create an executable program, it will run over the Windows Sockets ABI - the WinSock.DLL dynamic link library file - from any vendor. You will not have to recompile or relink your source code to execute your applications over different vendors' WinSock.DLL. Binary portability gives you the ultimate in flexibility and convenience.

1.72. What is the difference between COM and DLL?

A DLL (Dynamic Link Library) is an executable code packaged in a .DLL file, not part of the program that requires them. It could be simply a collection of much used functions that you want to share amongst programs or something more complex.

COM (Component Object Model) is not a physical "thing" like a DLL, it is a protocol for communication of objects. COM compliant objects expose properties that allow other programs to access/query/modify them. COM is an evolution of OLE (Object Linking and Embedding). OLE is a technology developed by Microsoft that allows embedding and linking to documents and other objects.

DLL's are basically used to expose functions and/or other resources like bitmaps. COM is a binary level specification for building components. You can use DLL to export your COM classes. COM components can also be exported as EXE files or Windows Service.

1.73. What is the difference between SDK and JDK?

A Software Development Kit (also known as an SDK or a devkit) is a set of development tools. It allows for applications to be created for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or any platform similar to

any of those listed. SDKs range from anything as simple as an API in the way that some files interface to a particular programming language or include sophisticated hardware in order to communicate with a certain embedded system. Some of the more common tools found in an SDK include debugging aids and similar utilities that are presented in an integrated development environment (or IDE).

The Java Development Kit (or JDK) is the most widely used SDK on the market. Developed by Sun Microsystems for Java developers, the JDK is a free software that was released under the GNU General Public License (or GPL). There is a plethora of components that make up the JDK. These components are a selection of programming tools. They include, but are not limited to java, the loader for all Java applications that interprets and is able to interpret the class files generated by the javac compiler; javac, which is the compiler that converts source code into Java bytecode; javaws, which is the Java Web Start launcher for the JNLP applications; jmap, which is an experimental utility that outputs memory map for Java and is able to print shared object memory maps or heap memory details of a given process; and VisualVM, which is a visual tool that integrates several command line JDK tools and lightweight performance and memory profiling capabilities.

SDKs include sample code and technical notes or other documentation that support this code in order to aid in the clarification of points from the primary reference material. Usually a software engineer receives the SDK from a target developer. SDK is thusly quite easily downloaded from the internet. Many SDKs are free of charge –mostly to encourage developers to use the system or language. They may have attached licenses in order to make them unsuitable for building software that is intended to be developed under an incompatible license. An SDK developed for an OS add on (QuickTime for Mac OS, for instance) may include the actual add on software itself for development use –if it is not to be redistributed.

The JDK is an extended subset of an SDK. Sun acknowledges under the terminology, the JDK is the subset of the SDK which is responsible for writing and running Java programs. What remains of this SDK is made up of extra software (Application Servers, debuggers, and documentation).

In summary:

1. An SDK is a set of development tools that allow applications to be created for certain software packages or platforms; the JDK is the most widely used SDK and is an extension of the SDK responsible for writing and running Java programs.
2. An SDK includes sample code and technical notes or other supporting documentation; the JDK includes components that are a selection of programming tools.

1.74. What is the difference between LDAP and Database?

Lightweight Directory Access Protocol (also known as LDAP) is an application protocol. This protocol is used specifically for querying data as well as modifying said data. This is performed by using directory services – that is, a software system that stores, organizes, and provides access to the information that is in a directory– running through a TCP/IP. The main function of any directory is to act as a set of objects with logically and hierarchically organized attributes –such as the telephone directory.

A database is simply a collection of data that has one or more uses. There are few ways in which a database is capable of being classified. One of the most common is classifying the data in terms of the type of content is listed –for example, bibliographic, full text, numeric, or image. Another way in which a database can be classified is according to an examination of database models or database architectures. This is accomplished by specific software organizing the data in the database according to said database model. The most common database model is that of the relational model – which is a database model based on first order predicate logic.

An LDAP session is instigated by a client. He accomplishes this by connecting to an LDAP server –this server is known as the Directory System Agent (or DSA). It is on the TCP port 389 by default. After the client has connected to the LDAP server, he sends an operation request to that server and in return the server sends a response (or number of responses). The client, however, does not have to wait for a response in order to send the next request –except in some cases. The server may, conversely, send the responses in any order. The server is also capable of

sending “Unsolicited Notifications” –meaning responses that are not responses to any request (before the connection times out, for example).

There are various database architectures that exist, and, in fact, many databases use a combination of strategies to function. Databases are comprised of software based “containers”. These containers are designed specifically to collect and store information in order to give users the power to retrieve, add, update, or remove the information automatically. Database programs are specifically designed to also give users the ability to add or delete any information necessary. Databases are usually in a tabular structure –meaning they consist of rows and columns.

Summary:

1. LDAP is an application protocol which queries and modifies data by using directory services; a database is a collection of data with one or more uses.
2. LDAP sessions are instigated by clients who connect to the LDAP server; there are various database architectures which many databases use in concert with one another.

1.75. What is the difference between Class and Object?

Object Oriented Programming, or OOP, is a very popular style of programming, due to its ability to handle more complex applications with a lot more code. This is because it organizes the data into objects that are comparable to real life objects. Class and object are two terms that are commonly used in OOP. In its most basic form, objects are the instantiation of classes.

In order to use objects in a program, you need to declare the properties and procedures in a class. For us to visualize this better, let’s discuss this using an example. If you want to create a program that deals with vehicles, you will need to create a class for vehicles. In the class you would create variables that will hold information relevant to vehicles. Values like passenger capacity, top speed and fuel capacity are typical along with procedures like start and stop. After creating the class for vehicles, you can now create objects in your application that are based on the vehicle class. You can create an object called car or motorcycle that is based on vehicles.

You can then fill up the relevant information on the object, and use it as you deem fit in your application.

As you already may have deduced from the example discussed above, the information that you will really use in an application are stored in the object, and not in the class. The class only defines the structure of the data, and what each procedure or function does.

Another excellent feature of classes is the ability to inherit properties and procedures from another class. Classes that inherit the properties of other classes are called subclasses. This shortens the work needed to define another class. If you want to define a class specific to cars, you can simply inherit the properties and procedures in the vehicle's class, as all cars are vehicles and will exhibit the same characteristics. The same is not done with objects, as there is really no practical use for globally inheriting an object's data. The basic practice for programmers is to create a subclass, and create the object from the subclass.

Summary:

1. An object is an instance of a class.
2. You define all the properties and functions in a class, while you use them in an object.
3. Classes do not hold any information, while an object does.
4. You can create subclasses, but not sub-objects.

1.76. What is the difference between ZIP and GZIP?

ZIP and GZIP are two very popular methods of compressing files, in order to save space, or to reduce the amount of time needed to transmit the files across the network, or internet. In general, GZIP is much better compared to ZIP, in terms of compression, especially when compressing a huge number of files.

Software that use the ZIP format are capable of both archiving and compressing the files together. These are two separate processes. Compression reduces the size of the file with the use of algorithms, while archiving combines multiple files, so that the output is a single file. GZIP is

purely a compression tool, and relies on another tool, commonly TAR, to archive the files.

It might seem like a minor thing, but it can affect the experience of the user in certain instances. The common practice with GZIP, is to archive all the files into a single tarball before compression. In ZIP files, the individual files are compressed and then added to the archive. When you want to pull a single file from a ZIP, it is simply extracted, then decompressed. With GZIP, the whole file needs to be decompressed before you can extract the file you want from the archive. When pulling a 1MB file from a 10GB archive, it is quite clear that it would take a lot longer in GZIP, than in ZIP.

GZIP's disadvantage in how it operates, is also responsible for GZIP's advantage. Since the compression algorithm in GZIP compresses one large file instead of multiple smaller ones, it can take advantage of the redundancy in the files to reduce the file size even further. If you archive and compress 10 identical files with ZIP and GZIP, the ZIP file would be over 10 times bigger than the resulting GZIP file.

Although both can be used with almost any operating system, each is popular in certain systems. ZIP is very popular with the Windows operating system, and has even been incorporated into the features of the OS itself. GZIP has a large following in the UNIX-like operating systems, such as the many Linux distributions.

Summary:
1. GZIP can achieve better compression compared to ZIP.
2. ZIP is capable of archiving and compressing multiple files, while GZIP is only capable of compression.
3. You can easily extract individual files from a large ZIP file, but not from a GZIP tarball.
4. ZIP is fairly popular on Windows, while GZIP is more popular on UNIX-like operating systems.

1.77. What is the difference between Abstract Class and Interface?

Abstract class (or type) is a type of in a nominative type system declared by the program. Though the name implies such, an abstract class may or may

not include abstract methods or properties. The distinction class refers to different language constructs that may be used to implement abstract types. Abstract classes can be characterized by a design issue that keeps with the best object oriented programming and by their unfinished natures.

An interface is an abstract type that classes must implement to specify an interface (generically speaking). Interfaces may only contain method signatures and constant declarations (both static and final), never method definitions. Interfaces simulate multiple inheritances and are used to encode similarities shared among various types of classes.

Abstract types are able to be created, signified, or simulated in several different ways. A programmer can signify abstract types by using the keyword abstract explicitly, by including one or more methods in the class definition, inheriting from another abstract type without overriding missing features necessary to complete the class definition, or by sending a particular method to the object oriented programming language known as this that does not implement the method directly.

Interfaces can be defined using abstract methods. Classes may also be implemented in interfaces. If a class does implement an interface and does not implement all its methods, the signifier abstract must be used, otherwise that signifier is not necessary (because all interfaces are inherently abstract). Classes can also implement multiple interfaces.

While interfaces are used to specify generic interfaces, abstract types can be used to define and enforce protocol (which is a set of operations which all objects that implement the desired protocol must support). Abstract types do not occur in languages without subtyping. As such subtypes are forced to implement all needed functionality, ensuring the correctness of program execution. There are several ways in which abstract types may be created: full abstract base classes are classes that are either explicitly declared to be abstract or contain abstract (unimplemented) methods; Common Lisp Object Systems include mixins that are based on the Flavors system; Java, of course; and Traits, which acts as an extension to Smalltalk.

Summary:

1. Abstract classes (or types) declare programs; interfaces are abstract types that all classes must implement in order to specify their interface.
2. Abstract types can be signified using the keyword explicitly; interfaces are inherently abstract, therefore do not need to be signified using the keyword at all (unless a specific class implements an interface but does not implement all its methods).

1.78. What is the difference between EXE and DLL?

The terms EXE and DLL are very common in programming. When coding, you can either export your final project to either a DLL or an EXE. The term EXE is a shortened version of the word executable as it identifies the file as a program. On the other hand, DLL stands for Dynamic Link Library, which commonly contains functions and procedures that can be used by other programs.

In the most basic application package, you would find at least a single EXE file that may or may not be accompanied with one or more DLL files. An EXE file contains the entry point or the part in the code where the operating system is supposed to begin the execution of the application. DLL files do not have this entry point and cannot be executed on their own.

The most major advantage of DLL files is in its reusability. A DLL file can be used in other applications as long as the coder knows the names and parameters of the functions and procedures in the DLL file. Because of this capability, DLL files are ideal for distributing device drivers. The DLL would facilitate the communication between the hardware and the application that wishes to use it. The application would not need to know the intricacies of accessing the hardware just as long as it is capable of calling the functions on the DLL.

Launching an EXE would mean creating a process for it to run on and a memory space. This is necessary in order for the program to run properly. Since a DLL is not launched by itself and is called by another application, it does not have its own memory space and process. It simply shares the process and memory space of the application that is calling it. Because of

this, a DLL might have limited access to resources as it might be taken up by the application itself or by other DLLs.

In summary:

1. EXE is an extension used for executable files while DLL is the extension for a dynamic link library.
2. An EXE file can be run independently while a DLL is used by other applications.
3. An EXE file defines an entry point while a DLL does not.
4. A DLL file can be reused by other applications while an EXE cannot.
5. A DLL would share the same process and memory space of the calling application while an EXE creates its separate process and memory space.

1.79. What is the difference between LIB and DLL?

When developing software, we are often asked whether we want to use LIB or DLLs in containing functions for the application. LIB is a static library where functions and procedures can be placed and called as the application is being compiled. A DLL or Dynamic Link Library does the same function but is dynamic in a sense that the application can call these libraries during run-time and not during the compilation. This presents a few significant advantages compared to using LIB.

For starters, you would have a single file that is significantly bigger as it contains all of the code while you would have multiple smaller files when using DLL. Compiling your functions and procedures would also allow you more reusability as once you are happy with the functions on the DLL because you can keep it as is with each version of the application and not have to mess with it. You can also use the same DLL if you want to create another application that uses the same functions and procedures. You can directly link to the DLL rather than copy the code from the source as you would need to do with LIB.

A problem with DLL is when you change the content of the DLL. This can lead to versioning problems where an application uses the incorrect version of the DLL causing problems. You need to keep track of your DLLs in

order to avoid these problems. You would not have this problem with LIB as you would only get one large file.

When developing the software and choosing DLL, you would still have a LIB file in your project. But unlike when using LIB, this file does not contain the code of the functions and procedures but only stubs that the program needs to call the procedures from the DLL's.

In summary:

1. A DLL is a library that contains functions that can be called by applications at run-time while LIB is a static library whose code needs to be called during the compilation
2. Using LIB would result in a single file that is considerable bigger while you end up with multiple smaller files with DLL's
3. DLL's are more reusable than LIBs when writing new versions or totally new applications
4. DLL files can be used by other applications while LIB files cannot
5. DLL's are prone to versioning problems while LIB is not
6. You would still have a LIB file when developing software with DLLs but it only contains stubs

1.80. What is the difference between Operating System and Kernel?

For most people, using a computer is second nature. This is made possible by the operating system that is running on top of the computer and hardware and makes it possible to communicate with it without having to use machine language or binary. The operating system provides us with an interface, whether graphic or text, where we can view the result of the commands we enter. It also provides us with an array of tools to configure the computer to our liking, at the very barest. But all this would not be possible without the kernel. The kernel is the core of the operating system and it is responsible for translating the commands into something that can be understood by the computer.

The aspect that a lot of programmers like about the kernel is in the abstraction. Hardware abstraction allows programmers to write code that can work on a wide array of hardware. Without hardware abstraction, each

program needs to be written specifically for a given hardware configuration and would probably not work in another. This is the case with device drivers. These are specific pieces of code that identifies the hardware and gives the operating system a means to communicate with the device.

Although the kernel is the core of the operating system, most people are not even aware of it existence because it is buried behind a lot of other software. To provide a whole package that lets people use their computers, an operating system includes software that covers a lot of the common uses of computers. This includes a simple word processing application and a media player among other things.

Applications are created for specific applications as it needs to interact with its kernel in order to communicate with the hardware beneath it. Since each kernel is different, applications for one operating system are not able to execute on others. Some applications are also not capable of executing in older or newer versions of the same operating system due to the changes that are implemented.

Summary:

1. An operating system is a software package that communicates directly to the computer hardware and all your applications run on top of it while the kernel is the part of the operating system that communicates directly to the hardware
2. Though each operating system has a kernel, this is buried behind a lot of other software and most users do not even know it exists

1.81. What is the difference between MFC and Win32?

The Windows API (Application Programming Interface) has come to be commonly known or referred to as Win32. If you want to create a program that would work in a Windows environment, you would need to have something that is compatible with Win32. MFC or the Microsoft Foundation Class is a class library in C++ that encapsulates certain portions of the Windows API in order to make it easier for programmers to build lightweight code.

Building an application for Win32 means that you would have to use its SDK in order to maintain compatibility and avoid glitches or any other problem. The problem with using the Win32 SDK is that you would need to manually write code for everything. This can lead to errors in code which can either be minor and quick to fix or major and be a headache to trace. The MFC is composed of functions that are most commonly used by programmers like creating windows or opening dialog boxes. Using the MFC reduces into a single line of code what would otherwise be composed of 10 or 20 lines making it simple and much quicker to build. Troubleshooting would also be a lot easier with the MFC since you would not need to delve into the actual coding of each function and you would only need to concern yourself about how you called the function.

The MFC also deals with the Windows environment directly, meaning that you would not need to concern yourself with the specific settings that each user has on his computer. This ensures that your program would appear correctly in most cases when using the MFC.

The MFC has been a very successful library that other programming languages have developed their own or adapted the use of the MFC for their own. Regardless of which framework you utilize, you would still be using Win32 if you intend to create programs for the Windows operating system. MFC just makes it easier and quicker for C++ programmers.

In summary:

1. Win32 is also known as the Windows API while the MFC is a C++ class library that wraps parts of the Windows API
2. The MFC consists of the most common operations used in building a Win32 application
3. Using the MFC makes the coding lighter and a lot simpler than using the Windows API directly
4. The MFC allows C++ programmers to use the current Windows environment.

1.82. What is the difference between GZIP and TAR?

The .tar.gz extension is a very popular one when downloading files especially in non-Windows operating systems. But unlike most extensions,

this is not meant to identify a single program that would be used to open this file; it points to two. Tar is a file format but is also the name of the program that produces the file. Tar is an archiver, meaning it would archive multiple files into a single file but without compression. Gzip which handles the .gz extension is the compression tool that is used to reduce the disk space used by the file.

Most Windows users are used to having a single program compress and archive the files. Some of the programs that do this are Winrar, 7zip, and Winzip. But this is not the case in the UNIX and Linux environments where archiving and compressing are two different operations altogether. Tarball is the name used to identify any collection of files that have been archived into a single file by the Tar application whether it is compressed or not. Although Gzip is the most common compression tool that is used with tarballs, it is not the only one. There is also another compression tool called Bzip2 that could compress the file further but would take a lot longer.

There are advantages and disadvantages in creating a zipped tarball compared to the standard applications that does both. A zipped tarball could have dramatically reduced filesize the compression tool is not compressing the files individually but treating the whole tarball as one big file. This is even more apparent when dealing with semi compressed files like GIF and JPG files.

The disadvantage of using this format is that you would be unable to extract a single file. Since the entire archive is compressed as a whole, you would also need to uncompress the whole archive before extracting a single file. Extracting a single file from a large zipped tarball could take significantly longer compared to extracting a single file from the same files compressed and archived by a different tool like Winrar or Winzip.

Summary:

1. Gzip is a compression tool used to reduce the size of a file
3. Tar is an archiver used to to combine multiple files into one
4. Gzip and Tar are usually used in tandem to create Tarballs that are compressed significantly

5. Extracting an individual file can take a lot longer with zipped tarballs than with other formats

1.83. What is the difference between add-on and plug-in?

Whenever we buy software for our computers, we always want the one that has the most features that are already included. But oftentimes the problem with software is either they do not have the right functionalities that you are looking for or they have too many features that it is impossible to find the one that you are looking for quickly. In order to customize the look and feel of their programs, software makers have allowed the use of plug-ins or add-ons.

Plug-in and Add-on are two terms that are pointing to the same functionality; they are simply extensions that extends the usability of the program. It just depends on the software maker on what to call the software extensions of their programs. These extensions could be made by other companies, individuals, or by the software makers themselves.

Plug-in is the term that is usually used when referring to third party software that is meant to interact with a certain program. Take for example your web browser; you would need to install a plug-in called flash player in order to play videos. Flash player is not native to any browser but is made by a separate company altogether. It is also compatible with all of the popular web browsers like IE, Firefox, and Opera.

An Add-on also extends the functionality of a certain program but they are usually meant to function on a certain program. Taking the web browser for comparison, add-ons that are meant for Firefox would only work with Firefox and so would for other browsers. These are usually not full blown software but are simply pieces of code that you can use to modify the interface. The most common add-ons for browsers are toolbars which take a little bit more space and give you instant shortcuts to certain online services. Add-ons are also very prominent in online games like World of Warcraft, where players who have a little know-how can create their own add-ons to help other players.

The separation between an add-on and a plug-in is not really that clear. They are both made to do specific functions that are suited to a certain user's preference. The primary reason why these codes are not embedded into the program in the first place is that they are not really that essential and while some people might appreciate having that, others might not and find it a nuisance. These are also tools that a software maker can utilize to motivate the members of their community to get involved in improving the software.

1.84. What is the difference between AJAX and Silverlight?

In the competitive world of computer software, Microsoft has been the biggest giant. Despite its superiority in the software market, there are still certain niches that Microsoft does not have total control over and are populated by some of the smaller market players. One of these is held by Adobe and its flash player.

Microsoft has now begun to threaten Adobe and its flash player with the new introduction to its software arsenal; Silverlight. Although silverlight has been in development since 2006, it has only been previewed and released to the public within the last year. Microsoft has even faced a lot of criticism for not sticking to the standard and building their own methodology in silverlight. Despite its detractors, silverlight is probably going to become popular, due in part by its inevitable inclusion into future releases of Windows and Internet Explorer. Another excellent aspect of Silverlight is the fact that it is also included in the .Net family and can be coded with any of the programming languages that are included with it.

Another good aspect of Silverlight is the fact that it can utilize AJAX (Asynchronous Javascript and XML) to create content that are even more dynamic than what is normally expected of an animation. Silverlight can request for data using AJAX even after it has fully loaded. Coupled with the excellent graphical controls, Silverlight could provide a seamless interface for most data driven transactions that up to now are still mostly in HTML or other related software.

Silverlight is said to be compatible to all Windows operating system that are running their latest Internet Explorer software. It is also compatible with OS

X, the operating system of Apple deployed in their Macs. And in a move that determines how much Microsoft wants Silverlight to succeed, Microsoft has partnered with Novell to create a version that is compatible with Linux named Moonlight. Even in the mobile industry, Microsoft is set to challenge Adobe with its forthcoming release of Silverlight for its own Windows Mobile 6 and even the Symbian platforms.

Despite coming under fire for its seemingly underhanded tactics in not following industry standards, Microsoft has posed Silverlight to be a strong market player. With its very wide deployment targets and support for technologies like AJAX that extends its capability even further than its established competitors, Silverlight might just prove be a solid piece of software and live up to its hype.

1.85. What is the difference between Winzip and WinRar

Compressing files have been a very essential process some time ago. When hard disk space was prime realty and buying extra hard disk drives is way over budget. Compression allowed people to store more in less space. Zip and rar are two of the most well-known compression formats in the world, with the former being the leader. With the introduction of windows, the makers of the software that compresses into these formats created GUIs for windows. Thus, winzip and winrar were born. Winzip was first on the scene and has ever since enjoyed its overall popularity.

Winrar is relatively new compared to winzip but it has proven itself to be quite a performer in compressing files, regularly outperforming its rival in almost every file type existent. It was also the first to introduce features that extended its capabilities, including context menu integration and archive spanning. Even if it offered several more advantages compared to winzip, the rar format did not overtake winzip in terms of general use. It was only very popular among the intellectuals and the people who really had great disk space problems.

In today’s world, hard disk space has really gone up exponentially, and disk space is no longer a concern for most people. And even if you manage to outgrow your current capacity, the cost of new drives with insanely high capacities is so low. There are still people who compress files to reduce

their size but not to save disk space; it is utilized for faster uploading or downloading to the internet. File spanning has also become one of feature that is more often used. Cutting a large file into more moderately sized chunks make it easier to email or upload files.

But aside from all the techy stuff, the primary use of compression software for the general public is to combine multiple files into one big file. This is for easy identification and transfer of a group of documents. This is usually done when submitting reports or pictures or anything that might be too many to handle individually. This is where winzip has its edge, simply because of its popularity. Passing files in winzip gives you a much higher chance that the other person can open the archive that you sent.

You should evaluate your needs if you want to choose between the two. If you really like performance and the added features you should pick winrar since it could also create zip archives. But if you want a no hassle path to creating archives that you can send to your boss then winzip is for you.

1.86. What is the difference between Windows and Linux?

Windows and Linux are both operating systems that have been developed to allow the utilization of computer resources. These two systems have a number of differences and require different things from users for proper operation of the system. Among the major differences is that Windows is a commercial operating system, while Linux is an open source operating system.

Windows is the widest used PC operating system today. It’s graphical user interface allows users with no programming experience or knowledge to navigate the system and complete the tasks. Some users have experienced stability problems, but with typical individual use the system has proven to be stable enough to prevent a massive shift. Windows OS has different versions that are tailored for various needs.

Linux is an open source operating system. It is renowned among users as being far more stable than the Windows system. Some network administrators and programmers prefer the system, but for a number of reasons the system has not gained the level of market penetration that

Windows or MacOS have reached. Little support by software makers has been among the problems. Though, with some knowledge, it is possible to run most Windows versions of programs on Linux systems. Also, while the system offers a graphical interface, there remains a bit of coding involved in the implementation of some programs. This can prove difficult for some users.

Linux and Windows have both proved they are operating systems that do not intend to go away. Windows consistently creates a system that is designed with individuals that have no computer or programming knowledge in mind as well as businesses and other commercial users. Linux continues to open its source code to anyone interested in improving the code and using the system and that makes it a favorite of many programmers.

1.87. What is an Object?

It combines both data structure and behavior in a single entity. Objects can be concrete, such as a file in a file system. Understanding the concept of an object is paramount to grasping object-oriented programming in Java. The three important concepts when describing an object are:

1. An object is not the same as a class. Classes are the blueprints for a part of an application that defines methods and variables. Objects are the individual incarnations of the class, sometimes called instances of a class. Two instances (objects) of the same class may contain different data, but will always have the same methods (behavior).
2. Objects are used in Java to bundle both variables (attributes) and methods (behavior) together into a single entity. Variables are used to hold the state of an object. For a person object, this may include the person's age, height, weight, etc. Methods are Java functions that define the behavior of the object; what it is allowed to do like change their height, weight or birth date.
3. Object-oriented technologies provide programmers and software developers many benefits. When using an object-oriented programming language like Java, you use abstract entities (software objects) to model real-world objects you find in everyday life. These real-world objects share two characteristics: they all have state and they all have behavior. For example, a program might use software

objects to model real-world bank accounts, automobiles, employees, or even a window on your computer.

1.88. What is a Class?

Class is a logical definition of a particular type of object. The class indicates which properties and methods a particular type of object provides. When defining a class, ensure that you do not use the term class and object interchangeably. A class is a blueprint, or prototype, that lists all methods and variables that go into defining, or constructing attributes common to all objects of a certain kind. Just like a blueprint, a Java class has exact specifications that will define its state and it behavior. The definition of a class is writing to a file with a .class extension. Consider creating an employee class. After creating the employee class, you must instantiate it (create an instance of it) before you can use it. When you create an instance of a class, you actually are creating an object of that type and the Java Virtual Machine (JVM) allocates memory for the instance variables declared by the class.

1.89. What is the difference between a class and an object?

A class is the logical definition of a particular type of object. An object is the instantiation of a class.

1.90. What is OOP?

Object-oriented programming (OOP) refers to the use of derived classes and virtual functions.

1.91. Is there anything you can do in C++ that you cannot do in C?

No. There is nothing you can do in C++ that you cannot do in C. After all you can write a C++ compiler in C.

1.92. What are the two things that make up an object in Java?

Its attributes (variables) and behavior (methods).

1.93. What is an abstract method?

A method defined in an abstract class which has no code associated with it. Its purpose is to serve as a placeholder, indicating that subclasses must have a concrete implementation for it. Abstract methods are created by labeling it with the keyword abstract.

1.94. What is an abstract class?

A class in Java must be declared as an abstract class if it contains one or more abstract methods. An abstract method can be thought of as a prototype method - it is a method that contains no body. It is nothing more than a signature method followed by a semicolon. The only way to use an abstract method is to subclass it and implement all of its abstract methods. One thing about an abstract class is that it can contain non-abstract methods and ordinary variable declarations. Keep in mind that not all methods in an abstract class have to be implemented in a single subclass. It is possible to subclass an abstract class without implementing all of its abstract methods, but the superclass must be declared as abstract.

1.95. What is Encapsulation?

Encapsulation is a technique used to control access to members of a class. Object oriented programming languages, including Java, provide the ability for one object to choose whether or not to expose its variables to other objects and allowing them to inspect and even modify its variables. An object can also choose to hide methods from other objects, forbidding those objects from invoking its methods. Using encapsulation techniques, an object has complete control over whether other objects can access its variables and methods even going as far as saying which other objects have access.

1.96. What are the benefits of Encapsulation?

There are two major benefits in encapsulating related variables and methods into a convenient software bundle. The first is modularity. This allows the developer to write and maintain source code for one object independently of source code for another object. The primary benefit of modularity is that objects can be effortlessly passed around in a system - providing developers to share objects with expected behaviors. The second benefit of encapsulation is information hiding. This allows the developer to write

classes with public interfaces that allow other objects to communicate with it. But the object can also maintain private information that is not accessible from other objects. This allows internal (private) methods to be changed without affecting other objects that depend on it. Users of your object do not have to know the internals of how your object works in order to properly use it.

1.97. What is Inheritance?

Inheritance is a mechanism that enables one class to inherit both the behavior (methods) and the attributes (variables) of another class.

The term “Base Class” is used to describe the class used as a basis for inheritance. Other terms used to describe the base class are “Superclass” and “Parent Class”. The term “Derived Class” is sometimes used to describe a class that inherits from a base class. Other terms used are “Subclass” and “Child Class”.

The extended class has all the features of the base class, plus some additional features. Consider for example, a Student class that is derived from a more general People class. The Student class can add a field called salary, and jobPosition that the People class lacked.
The use of inheritance enables developers to add features to an already existing class and is considered an important aid in the design of the program that has many related classes. This makes it easy for developers to reuse classes which is a key benefit of object-oriented programming design.

1.98. What is Polymorphism?

Polymorphism lets an application decide at runtime, every time an operation is called, which of several identically named methods to invoke. Polymorphism, sometimes referred to as dynamic binding, late binding or run-time binding is a property of Java and other object-oriented software by which an abstract operation may be performed in different ways in different classes. It primarily deals with type hierarchies. In order for polymorphism to work, these different classes must be derived from the same base class. When using polymorphism in practice, the developer will involve a method call that actually executes different methods for object of different classes.

This allows developers to write code that does not depend on specific types, thus simplifying and clarifying program design and coding.

1.99. What are Virtual Functions and are they supported in Java?

Yes, virtual functions are supported in Java. In fact, Java class functions are virtual by default. A virtual function is a member function that is expected to be redefined in a derived class. When referring to a derived class object using an object reference to the base class, you can call a virtual function for that object and execute the derived class's version of the function. Within Java, the programmer does not need to declare a method as virtual. This is the default behavior in Java.

Working with a non-object-oriented language, when the compiler comes across a function (or procedure), it determines precisely what target code should be called and builds the machine language to represent that function call. In an object-oriented language like Java, this is not possible. The correct code to invoke is determined based upon the class of the object being used to make the call, not the type of the variable. The byte code is generated so that the decision of which method to invoke is made at runtime.

Virtual functions enable runtime binding of function calls to the actual function implementation. From a technical perspective, virtual functions are functions of subclasses that can be invoked from a reference to their superclass.

1.100. What is runtime binding?

Within the context of Java, runtime binding means that the resolution of a function call occurs at runtime. Meaning, when you call a (virtual) function, the call is not resolved until at runtime by the JVM.

1.101. What are Web services?

In recent years, Microsoft, IBM, and other technology vendors have been working to institute a more formalized infrastructure for accessing and publishing services or processes (through remote procedure calls, or RPCs)

over the Internet or a corporate intranet. These initiatives have resulted in what are now known as Web Services.

A Web Service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols. More precisely, a Web Service is defined by the World Wide Web Consortium (W3C) as follows: “a software application identified by a URI [Uniform Resource Identifier] whose interfaces and bindings are capable of being defined, described, and discovered as XML artifacts.”

Web services are a new breed of Web application. They are self-contained, self-describing, modular applications that can be published, located, and invoked across the Web. Web services perform functions, which can be anything from simple requests to complicated business processes. Once a Web service is deployed, other applications (and other Web services) can discover and invoke the deployed service.

Web services represent a new and simplified standards-based model for creating and connecting distributed applications across the internet. They are built around widely-adopted, existing internet standards (such as TCP/IP, HTTP, and XML) and a set of new standards, which include Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery, and Integration (UDDI).

The Web Services technology stack consists of the following layers:

- UDDI - The UDDI specification enables businesses to quickly, easily, and dynamically find and transact with one another. UDDI allows a business to:
  - Describe its business and its services
  - Discover other businesses that offer desired services
  - Integrate with these other businesses.
- WSDL - WSDL is an XML (XML 1.0) format for describing Web Services. WSDL enables one to separate the description of the abstract functionality offered by a service from the concrete details of a service description, such as how and where that functionality is offered. Simply put, WSDL defines the Web Services’ interface,

providing the initial point of contact for an external application or component.

- SOAP - SOAP provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized and distributed environment by using XML. It consists of a framework for describing the data in a message and how to process it and for encoding data types used in remote procedure calls.

IT organizations have realized that Web Services present a tangible solution to many of their most pressing challenges. Specific benefits of Web Services include the following:

- Reduced integration complexity - Web Services provide greater flexibility and adaptability when integrating disparate systems.
- Increased component reuse - Numerous infrastructure vendors are providing mechanisms for publishing existing software components as Web Services. This approach allows organizations to maximize the value of legacy applications while decreasing total development costs.
- Improved interoperability - Microsoft and the J2EE community are focused on ensuring interoperability between .NET and J2EE using Web Services.
- Migration toward a service-oriented architecture (SOA) - While Web Services do not guarantee an SOA, leveraging Web Services for new development begins to instill the discipline required for an SOA.

1.102. Describe Web Services development platforms.

To build web services, two main development platforms are in use today: the .NET platform from Microsoft and the J2EE platform from the Sun community. The standards-based nature of Web services ensures that Web services built on either platform are interoperable.

A Web Service (commonly called a service) comprises some content, or some process, or both, with an open programmatic interface. Simple examples: currency converter, stock quotes, dictionary, travel planner, procurement workflow system.

A service has the following characteristics:

- Internet-based application that performs a specific task and complies with a standard specification.
- An executable that can be expressed and accessed using XML and XML messaging.
- Can be published, discovered, and invoked dynamically in a distributed computing environment.
- Is platform- and language-independent.

1.103. What is SOAP and what is SOAP used for?

SOAP is a simple XML based protocol to let applications exchange information over HTTP. Or more simply: SOAP is a protocol for accessing a Web Service.

SOAP:

- is a communication protocol
- is for communication between applications
- is a format for sending messages
- is designed to communicate via Internet
- is platform independent
- is language independent
- is based on XML
- is simple and extensible
- allows you to get around firewalls
- is developed as a W3C standard.

SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts:

- an envelope that defines a framework for describing what is in a message and how to process it,
- a set of encoding rules for expressing instances of application-defined data types, and
- a convention for representing remote procedure calls and responses.

1.104. Why SOAP?

It is important for application development to allow Internet communication between programs. Today's applications communicate using Remote

Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this. SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

1.105. Is SOAP secure?

SOAP is as secure as HTTP is. SOAP travels across HTTP and enters via port 80, as does HTTP. The only difference is that a SOAP message carries a payload of XML, whereas HTTP would normally carry a payload of HTML, JavaScript etc. SOAP messages are very clear about their intent and declare this in their headers. Your firewall can read these headers and decide what to do. Your application when it receives the message should then examine the SOAP message to make sure that the headers match what is in the payload. SOAP can run over HTTPS (SSL) with no problems.

1.106. What is WSDL?

WSDL is a language for describing the capabilities of Web services. Proposed by IBM and Microsoft, WSDL combines the best of IBM's NASSL (Network Accessible Services Language) and Microsoft SOAP Contract Language. WSDL is based on XML and is a key part of the UDDI initiative. The WSDL document specification helps improve interoperability between applications, regardless of the protocol or the encoding scheme. The WSDL 1.1 specification defines WSDL as "an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages."

Essentially, a WSDL document describes how to invoke a service and provides information on the data being exchanged, the sequence of messages for an operation, protocol bindings, and the location of the service. A WSDL document defines services as a collection of endpoints, but separates the abstract definition from the concrete implementation.

Messages and port types provide abstract definitions for the data being exchanged and the operations being performed by a service. A binding is provided to map to a concrete set of ports, usually consisting of a URL location and a SOAP binding.

1.107. How does CORBA compare to RMI? to SOAP? to .NET?

CORBA - is set of standards, definitions, language mappings, application interfaces, standard services. Many of Java interface standards are Java mapping of standard CORBA interfaces sometimes with small extensions. One part of CORBA is protocol description between server and client for interchanging data and calling procedure. This part of CORBA standard is called GIOP (General Inter-ORB Protocol) and TCP/IP realization is called IIOP (Internet Inter-ORB Protocol).

RMI - is protocol for remote invocation of services for Java and only for Java. As low level protocol is using proprietary communication and both sides (client and server) have to be in Java.

RMI with IIOP plug-in - it's standard RMI with IIOP used as communication low level protocol. Thanks to this you can have full interoperability between RMI and CORBA and IIOP plug-in is used as bridge between this two worlds. However if you want build heterogeneous, enterprise, stable solutions then I would advise you to use full CORBA and not intermix RMI with CORBA. Some CORBA implementations are free, tested, very stable, fast, well documented.

SOAP - Is XML based protocol for interchanging XML objects. It's description of wire format for interchanging XML objects. It give you answers on question similar to this: "How to transfer object 2,3 and 5 from my set on server to client".

.NET - something very similar to what already J2EE (Java Enterprise Edition) have except reason that .NET is only in state of strategy and promising great features.

1.108. What is Ant?

Apache Ant is a Java-based build tool. In theory, it is kind of like Make, but without Make's wrinkles.

1.109. Why another build tool when there is already make, gnumake, nmake, jam, and others?

Because all those tools have limitations that Ant's original author couldn't live with when developing software across multiple platforms. Make-like tools are inherently shell-based - they evaluate a set of dependencies, then execute commands not unlike what you would issue in a shell. This means that you can easily extend these tools by using or writing any program for the OS that you are working on. However, this also means that you limit yourself to the OS, or at least the OS type such as Unix, that you are working on.

Makefiles are inherently evil as well. Anybody who has worked on them for any time has run into the dreaded tab problem. "Is my command not executing because I have a space in front of my tab!" said the original author of Ant way too many times. Tools like Jam took care of this to a great degree, but still have yet another format to use and remember.

Ant is different. Instead of a model where it is extended with shell-based commands, Ant is extended using Java classes. Instead of writing shell commands, the configuration files are XML-based, calling out a target tree where various tasks get executed. Each task is run by an object that implements a particular Task interface. Granted, this removes some of the expressive power that is inherent by being able to construct a shell command such as

`find . -name foo -exec rm {}`,

but it gives you the ability to be cross platform - to work anywhere and everywhere. And, if you really need to execute a shell command, Ant has an <exec> task that allows different commands to be executed based on the OS that it is executing on.

1.110. How do you select which partition to choose from - Fat, Fat32, and NTFS for W2K servers?

It is important to remember that Microsoft advises that you always choose NTFS unless you will be dual-booting with an additional operating system like Windows 9x. Remember that FAT and FAT32 offer no file level security like in NTFS. And unlike FAT, NTFS also offers disk compression, disk quotas, and file encryption. When faced with a question asking either a) which file system should you choose if you want file encryption and disk compression or b) Which file system should you choose to dual boot, make sure you know the differences listed above and choose the appropriate answer.

If you decide to use FAT or FAT32, setup will decide how the partition is formatted by looking at the size of that partition. A partition smaller than 2 GB will be partitioned as FAT, while one bigger than that will be setup as FAT32. If you require a dual boot scenario, only the system partition has to be FAT or FAT32, and all other drives can be NTFS. (On a side note, Windows 2000 can be installed on a 4 GB FAT partition. However, this can only be done in W2K and not in DOS.) Additionally, you can convert from FAT to NTFS at any time by using the convert utility at the command prompt, and the data on the drive will be preserved. You cannot change back from NTFS to FAT without data loss.

1.111. Why use coding/programming standards or conventions?

Programming standards are implemented and enforced by progressive, quality conscience companies because they:

- Promote consistency, making code easily understood and supported by others
- Assure easier maintenance
- Promotes high quality, reliable code
- Provide a framework/baseline for quality improvement practices
- Promote portability across platforms, operating systems, compilers
- Streamline the code review process
- Promote good coding practices
- Are generally a part of ISO 9000, CMM SPM quality initiatives.

Code conventions are important to programmers for a number of reasons:

- 80% of the lifetime cost of a piece of software goes to maintenance.

- Hardly any software is maintained for its whole life by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.

1.112. Why use an automated programming standard review tool?

Reviewing source code automatically, rather than manually (or not at all), has the following direct benefits:

- Reduced development, maintenance, and code review time
- Eliminates requirement for software development resources to perform tedious review tasks manually
- Allows enforcement in a controlled fashion, providing an iterative, selective, or evolving style of enforcement
- Provides much higher accuracy than manual review methods
- Allows collection and monitoring of software metrics
- Helps to identify previously undetected bugs
- Allows automatic adherence to evolving ANSI/ISO standards
- Provides a foundation for improvement of company programming standards
- Helps reduce number of compiler iterations
- Eliminates need for memorization of programming standards
- Eliminates need to make continuous references to standards document.

1.113. What is the difference between fault-tolerance and redundancy?

The goal of redundancy is, by using duplicated equipment, to improve the availability of the system. Here primary will be active and secondary will be idle. However the goal of fault-tolerance is, by using duplicated equipment, to improve the availability of system and also to eliminate bad signals going to the field due to hardware failure. Here both the primary and secondary will be processing logic under normal conditions.

The problem is “how to make a system more reliable.” The solution is through the use of fault-tolerance. There are different approaches to fault-tolerance:

- make the hardware more reliable and resistant to many faults

- use duplicate hardware to back-up parts of the system
- use software and systems approaches to bypass failed system elements
- all of the above.

1.114. When do you use radio button and checkboxes?

Radio buttons are used when you need to select only one of the many choices available while checkboxes is used when the user need to select one or more than one of the many choices available.

1.115. What are your company's most important criteria for selecting a PC vendor?

Criteria for selecting a PC vendor include:

1. Cost and value
2. Customer service
3. Reliability and product quality
4. Proven technology
5. Vendor and product reputation
6. Performance
7. Ability to buy direct
8. Service agreements and guarantees
9. Delivery
10. Industry standards
11. Ease of administration
12. Customized products
13. Advanced technology solution
14. Projected ROI
15. Scalability

1.116. Do you think technology standards are more important than the technologies themselves?

Technology standards are more important than the technologies themselves, because the standards make technologies more accessible. The success of a technology depends in part on the need for that technology, but it also depends on how accessible it is. Open standards make technology accessible and provide other benefits as well.

1.117. What is Serialization?

Serialization is the process of making objects persistent. A persistent object is one that can save its own state when a program terminates and then can restore its state when the program begins execution again.

1.118. What is a Control?

A Control is like a small Windows program.

- Controls have to be constructed. When created, controls are given a position and certain styles.
- As windows, controls process event messages. Usually, these events cause some internal change to the control. For example, a button will appear depressed, or a string in a list box will appear high-lighted.
- As objects, controls have member functions. Control member functions can be used to manipulate the control objects. For example, we can load a list box with some string values.

1.119. What is a resource?

A resource is anything other than code that is part of your program. This definition does not include data that your program might process. The two most notable program resources are menus and dialog boxes.

1.120. What is a Virtual server?

Virtual server lets multiple operating systems run on a single physical machine.

1.121. What does LAMP stands for?

LAMP stack of open source infrastructure software includes the Linux server operating system, Apache Web server, MySQL database and PHP Web development language.

1.122. After the release, which are the biggest points of pain in your deployment process?

In the decreasing order of pain:

1. Integration
2. Upgrade

3. Configuration parameters
4. Deployment architecture
5. Performance
6. Installation

1.123. What is the difference between a Web developer and a Web designer?

A Web developer strictly builds and maintains websites. However, a lot of people who create a site from start to finish - designing graphics and Web pages, figuring out the site map, then producing the site - call themselves Web developers as well. Typically, those who conceptualize and plan out the site structure are Web designers. Developers use some form of HTML to build the actual pages. A Web developer's other responsibilities could include optimizing include optimizing graphics for the Web and producing rich media such as Flash, streaming media, or online audio.

1.124. What do physicians use IT for?

Physicians use IT for the following applications in the decreasing order of importance or popularity:

1. Billing/claims submission
2. Patient appointment reminders
3. Lab orders/results
4. Communication with hospital
5. Claims status
6. Patients' records
7. Patients' eligibility
8. Diagnostic imaging/radiology
9. Referrals
10. Procurement of supplies
11. Charge capture
12. Clinical protocols/pathways
13. Prescribing
14. Telemedicine

1.125. Describe the document management (DM) lifecycle.

1. Creation: Document gets generated and entered into the system (data entry).
2. Reference: Document gets routed and revised. The DM system tracks who sees the file and any changes they make to it. Then, a second record gets created - a document tracking a particular document's life.
3. Historic: After about 120 to 180 days, the document is no longer subject to revision and becomes an actual record.
4. Archive: Now historic, the document gets stored in a database. For Sarbanes-Oxley, financial documents must be archived for seven years.
5. Destroy: No longer needed, the document gets erased from the system entirely - and securely.

1.126. What is document management?

Document management allows you to:

- Store your company's most important documents in one central repository, regardless of file type (IM, email, text file, etc.).
- Search for and retrieve documents by keywords.
- Create controls to secure documents and limit employee access.
- Automate routing of documents, speeding workflow.

1.127. When should you consider using document management system?

Document management tools can help your company comply with regulations imposed under the Sarbanes-Oxley Act. Work closely with your business units or division managers to get the maximum business value from your document management system. Consider using document management software if your company:

- Is traded publicly and must comply with federal regulations such as Sarbanes-Oxley.
- Has no way to track changes made at every stage of the document's lifecycle.
- Can make use of it beyond regulatory compliance issues to improve product life-cycle management and customer service.

Things to think about:

- Creating a records retention policy before buying software.

- Including your general counsel, CFO and internal/external auditors in discussions about data-use authorization, retention and destruction policies.
- Making sure your company is not already using a document management program before you buy a new one.

1.128. Compare various load balancing schemes.

A number of load balancing methods exist, and Network Load Balancing has unique advantages relative to each of them.

Round robin DNS: Round robin DNS is a popular solution for enabling a limited form of TCP/IP load balancing for Web servers. However, it does not function effectively as a high-availability solution. Round robin DNS uses DNS to map incoming IP requests to a defined set of servers in a circular fashion. In the event of a server failure, round robin DNS continues to route requests to the failed server until the server is removed from DNS, and even then many users must wait for DNS to time-out their connections before being able to successfully access the target Web site.

Hardware-based load balancing: Network Load Balancing and hardware load-balancers typically offer very similar functionality: both services transparently direct client requests for a single IP address to multiple hosts within a cluster. Hardware load-balancers generally use a technique called network address translation (NAT), that exposes one or more virtual IP addresses to clients and forwards data for the hosts by translating IP addresses and resending network packets. When a single hardware load-balancer is used, this technique introduces a single point of failure between the cluster and the clients. To provide high availability, a backup load-balancer is needed. Address translation and retransmission also imposes relatively higher overhead and delivers lower bandwidth than does Network Load Balancing's filtering technique.

Network Load Balancing has three distinct advantages over hardware-based load-balancers:

1. Network Load Balancing has significantly higher throughput.
2. Unlike hardware load-balancers, which have limited maximum performance, Network Load Balancing's performance automatically

improves with the speed of the cluster hosts and the local area network (LAN), ensuring that it will never become a bottleneck as LAN and host speeds increase. Network Load Balancing can easily accommodate a change to gigabit Ethernet without requiring any upgrade.

3. Network Load Balancing avoids a single point of failure, since it runs in parallel on all hosts in the cluster. Most hardware load-balancers require two hardware units to avoid a single point of failure, with the second equally expensive hardware component operating in a passive mode.

Dispatcher software load balancing: Software-based load balancing products that employ various dispatcher models for load balancing face many of the disadvantages found in hardware load balancing products. Dispatching techniques, whether implemented by network address translation or other methods (such as HTTP redirection), introduce higher overhead than does Network Load Balancing. This limits throughput and restricts performance. Also, the entire cluster's throughput is limited by the speed and processing power of the dispatch server.

These products require all incoming connections to be handled by one dispatch server, where they are then either processed or routed to other servers in the network. This introduces a single point of failure, that can only be eliminated by moving the dispatching function to a second server after a failure occurs. This recovery technique adds complexity and recovery time to the solution. The distributed software load balancing model used by Network Load Balancing avoids this problem by removing any dependency on a single server.

1.129. What is InstallShield?

Its features allow you to author MSI, InstallScript, and cross-platform installations, and extend them to configure database servers, Web services, and mobile devices with one, easy-to-use tool. InstallShield's all-in-one functionality provides developers a single, installation-authoring solution for every platform, operating system, and device.

1.130. Why do we model software?

To achieve a high level of quality in the final software product.

1.131. What are the three major steps to modeling software?

Analysis, design, and implementation

1.132. What is a activity diagram?

Used to analyze the behavior within more complex use cases and show their interaction with each other. Activity diagrams are very similar to statechart diagrams in that they represent a flow of data; however, activity diagrams are used to model business workflows during the design of use cases in the analysis phase of modeling, whereas statechart diagrams are used to represent objects in the design phase. Activity diagrams are usually used to represent the more complicated business activities, helping you identify use cases or the interaction between and within the use cases.

1.133. What are the steps to modeling an activity diagram?

Steps to model an activity diagram include:

1. Identify the business use cases to model.
2. Model the primary paths of each use case.
3. Model all alternative paths.
4. Add swim lanes when necessary.
5. Refine high-level activities into their own activity diagram.

1.134. What is a sequence diagram?

Sequence diagram is used to show interaction between actors and objects and other objects. Messages are sent from actor to object, object to object, and object to actor to show the flow of control through a system. Sequence diagrams are used to realize use cases by documenting how a use case is solved with the current design of the system. Sequence diagrams model the interaction between high-level class instances, detailing where process control is at every stage of the communication process. Sequence diagrams can be used to show every possible path through an interaction, or show a single path through an interaction.

1.135. What are the steps to modeling a sequence diagram?

Steps to model a sequence diagram include:

1. Decide on the workflows that you are going to model.
2. Lay out the objects for each model.
3. Include message and conditions for each model.
4. Draw a single, generic diagram if possible.

1.136. What are some differences between use case diagrams and sequence diagrams?

Use cases are generic, and explain to management and non-technical people what the system will be capable of doing. In many cases, use cases represent the requirements provided by management. Sequence diagrams model the steps that will be taken to achieve each use case within the system.

1.137. What is the difference between synchronous and asynchronous messages?

Synchronous messages force the system to wait until they are finished before executing the next message. Asynchronous messages allow parallel processing because they do not hold up other messages from being executed.

1.138. What are the steps used to create a class diagram?

Class diagram is used to represent the different underlying pieces (classes), their relationship to each other, and which subsystem they belong to. The steps used to create a class diagram are:

1. Find the classes and their associations.
2. Find the attributes and operations of each class.

1.139. What are collaboration diagrams?

Used to represent the interaction and the relationships between the objects created in earlier steps of the domain modeling process; can also be used to model messages between different objects, and how their advance associations can be used to navigate the model.

1.140. Why do we model a collaboration diagram?

To illustrate the communication between objects or roles.

1.141. What are the steps required to create a collaboration diagram?

1. Identify the elements of the diagram.
2. Model the structural relationship between those elements.
3. Model the instance-levl diagram.

1.142. What are the basic notational components of statechart diagrams?

States and transitions

1.143. What is the difference between statechart diagrams and activity diagrams?

Statechart diagrams model behavior that changes the state of a single entity, whereas activity diagrams model behavior among different entities.

1.144. What is the difference between events and actions?

Events are behaviors that cause states to change, whereas actions are behaviors that occur as a result of an event. Actions: Used to represent procedures that usually change the state of the system.

1.145. What is a statechart diagram?

Used to represent a single object and how its behavior causes it to change state. Very similar to a well-known state machine model, the statechart diagram is used during the crossover between the analysis and design phases. A statechart diagram is a wonderful way to visualize the flow of an application.

1.146. What are the steps involved in diagramming a statechart diagram?

1. Identify the entities that need to be further detailed.
2. Identify the start and end states for each entity.
3. Determine the events relating to each entity.
4. Create the statechart diagram beginning with the start event.
5. Create composite states where necessary.

1.147. What is a component?

A single piece of software. It can be a file, product, executable, script, or so on. It is a replaceable part of a system that packages the implementation of the realization of a set of interfaces.

1.148. What is a component diagram?

Used to illustrate how components of a system interact with each other. It would show dependencies between source files and classes as well as which components they belong to.

1.149. What is a deployment diagram?

Models where components will wind up after they are installed on systems and how the components interact with each other. Deployment diagrams are modeled to illustrate relationships between hardware.

1.150. What is the difference between a component and a deployment diagram?

Component diagrams illustrate relationships between pieces of software, while deployment diagrams illustrate relationships between pieces of hardware.

1.151. What do components represent?

Source code, binaries, executables, scripts, or commans files

1.152. What are the relationships between components called?

Dependencies

1.153. List the steps to create an implementation diagram.

1. Add nodes
2. Add communication associations.
3. Add components, classes, and objects.
4. Add dependencies.

1.154. What is meant by stereotyping dependencies?

A dependency can be stereotyped to give meaning to the relationship.

1.155. What is a MSI file?

Microsoft Installer (MSI). An .msi file is a database of all the files, settings, and configuration information for the associated application. When you install Office on your computer, the .msi file is saved in a hidden folder. Without this file, Windows Installer cannot update your configuration, install optional features, or apply software updates. Office cannot be installed, repaired, or updated if the .msi file is not found.

1.156. Why the Microsoft Installer (MSI) format?

For stability purpose and easier management. Have you ever uninstalled software, just to find out that it leaves a lot of files behind? Those left behind, not to mention those pesky registry keys, which too much often prevent updates and upgrades. The Microsoft MSI technology was developed to address those problems. The applications in .msi format are made of a database and include all the needed files and registry keys for the software to function properly. When MSI packages are well developed, they have the capacity of self-healing, working without conflict and uninstall completely to leave your PC's clean. Furthermore, only the applications in MSI format can be deployed and managed thru the GPO's of Microsoft Active Directory.

1.157. Describe severity levels.

Severity 1: The system is down or inoperative or major software problems exist that critically impact the customer's business.
Severity 2: The system is operational but in a degraded mode or a major software problem exists that causes significant impact to the customer's business.
Severity 3: The system is operational but is experiencing minor degradation or a minor software problem exists that causes minimal impact to the customer's business.
Severity 4: The customer requests product information, a product enhancement, or a new feature.

1.158. What is user management?

Creating and modifying users, access, roles, and responsibilities. Roles are used to give users or groups of users permission to perform certain tasks.

1.159. What are consoles?

Consoles are workstation-based applications that allow security professionals to perform day-to-day administrative and operation tasks such as event monitoring, rules authoring, incident investigation, and reporting.

1.160. What are managers?

Managers are server-based applications.

1.161. Describe the traffic entering and leaving the corporate network? Email content versus Web content.

A recent survey conducted by the American Management Association found that 60% of employers monitored email, however, only 4% of the information entering and leaving the corporate network was emailed. Here is the break-up:

Email: 4%
Other (includes encrypted traffic, telnet, ftp): 6%
Web-Traffic: 90%

1.162. How information leaves the corporate network?

Only 1.1% of the content leaving the corporate network was emailed leaving most corporations still vulnerable when employees webmail, hotmail, instant message or post confidential information.

1.163. What is leaving the corporation in email attachments?

Of all emails sent out of the corporate network, excel spreadsheets were the most popular attachment. Here is the break-up:

Excel: 36.2%
Zip: 16.2%
Jpeg: 11.2%

Pdf: 9.4%
Tiff: 6.6%
MS Word: 6.5%

1.164. Encrypted versus unencrypted content

Encrypted data ranked very low and consumed only 2.2% of bandwidth total. This means that efforts to encrypt databases and build encrypted tunnels to secure companies' sensitive data are not being used when information is sent or received by employees. Here is the break-up:

- Encrypted: 2.2%
- Unencrypted: 97.8%

1.165. What is safety and security?

Safety is protection from mistakes. Security is protection from malice.

1.166. What are the advantages of LDAP?

Deploying multiple applications within intranet or extranet environments creates business advantage but exacerbates security administration and weaknesses. To escape this dilemma, organizations are investing heavily in centralized access control security architectures. Lightweight Directory Access Protocol (LDAP) has emerged as a leading enabling standard for implementing centralized access control within corporate intranets and extranets.

LDAP Benefits:

- Easy administration - LDAP directories give administrators centralized access control. Various applications can be managed in one virtual location by storing user information such as security profiles, preferences, configuration information, personalization data, or other details in a centralized server.
- Single sign-on (SSO) - LDAP is also the cornerstone of single sign-on nirvana for application deployment. Through single sign-on, a user or application signs onto the authentication mechanism only once.

The directory identifies the person or application, reviews the corresponding profile, and allows access to all resources and functions for which the user is authorized. Users love single sign-on because it eliminates the need to log on individually to the various systems they must access during the workday.

1.167. What is SSO?

Single Sign On (SSO) eliminates one of the biggest security headaches today - maintaining multiple IDs and passwords. In addition, it decreases the workload on help desk staff while enhancing employee satisfaction and productivity.

Similarly to the way user provisioning provides streamlined access for a user, simplified access to business applications is enabled through single sign-on technology. From both an administrative and user perspective, single sign-on relieves the frustration of having to remember multiple passwords to access multiple systems. Implementing single sign-on functionality helps reduce the IT department's administrative overhead by providing one interface to manage multiple systems. Security breaches have become common around password theft and unauthorized access to systems, which has created a need for a strong authentication method to be implemented.

1.168. What types of security technologies have you used?

SSL (Secured Socket Layer), VPN (Virtual Private Network), Encryption/Decryption, PGP (Pretty Good Privacy) IPSec, UserID/Password, Digital Certificates, Firewalls, DMZ (demilitarized zone)

SiteMinder, LDAP, Single Sign On (SSO)

Information assurance employs a variety of technologies to provide the following services:

- Authentication - verifies the source of information
- Integrity - information received is what was sent
- Non-repudiation - the sender and receiver cannot deny involvement later
- Confidentiality - information denied to the unauthorized
- Availability - information when you need

1.169. What is authentication?

The process of identifying an individual that is usually based on a user name and password. Authentication verifies that the individual is the person he or she claims to be, but cannot verify the access rights of that individual.

1.170. Hash functions

Hash functions are among the most widespread cryptographic primitives, and are currently used in multiple cryptographic schemes and security protocols, such as IPSec and SSL. Their primary application is their use for message authentication, integrity, and non-repudiation as a part of the Message Authentication Codes (MACs) and digital signatures.

1.171. Describe what PKI is and how it works.

Public Key Infrastructure (PKI) is the trusted foundation for supporting user authentication and is used for encrypting data transmissions.

A user (such as a Web site) generates a key pair, public and private.
The user sends the public key to a Certificates Authority (CA), who creates a certificate for the user's public key and publishes it.
When a browser visits the site, it uses the site's public key to encrypt the data, which can only be translated using the site's private key.
Browser also generates its own public and private keys for each session.

There is a public key and a private key. Private key holder use private key to encode a document, only the public key can decode it. When a document

is decoded with public key, the public key holder is sure that it comes from private key holder. When the public key holder want to send a document to private key holder, he use the public key to encode the document, no other person can see it. Only the private key holder can use his private key to decode the document and read it.

PKI is the set of hardware, software, people, policies and procedures needed to create, manage, store, distribute and revoke Public Key Certificates based on public key cryptography. PKI helps ensure repeat business by protecting distributed user identity and allowing transactions to take place in a trusted environment.

authentication using PKI

Authentication is one of the most important objectives in information security. Public key cryptography is a common means of providing authentication. Some examples are X.509 and PGP. In the public key infrastructure, each user is associated with a public key, which is publicly available, and with a private key, which is kept secret by the user. A user signs something with his private key, and this signature can be authenticated using the user's public key.

The ability to exchange public keys securely is an essential requirement in this approach. Certificates are considered to be a good way to deliver public keys, and are popularly used in today's public key infrastructures. Intuitively, a certificate is an authority telling about a user's public key. In a hierarchical system, such as X.509, we usually assume that the certificates contain true information, because the authority is secured and trusted. In non-hierarchical system, such as PGP keyrings, each user becomes an authority. Such systems are referred to as webs of trust. With webs of trust, it may be risky to expect all certificates to contain true information, because not all users are fully secured and trusted. Accepting a false public key (that is, believing it contains true information) undermines the foundation of authentication.

1.172. What is access control?

The general solution to security design problems has always had two parts: (1) Trust the people you have to trust, but (2) make sure that they are who they say they are.

Proving who you are is technically called “authentication”. You have used authentication every time you have given a username and a password or plugged in a smartcard. Authentication proves who you are, and in most systems there follows some sort of one-for-one match between who you are and what you are allowed to do. The combination of authentication (who you are) and authorization (what you can do) is generally referred to as “access control”.

For many reasons access control can no longer meet either the security challenge or the economic challenge.

Access control does not scale up indefinitely. When you consider the access control model, you have what amounts to a matrix: One row for each person (or thing) that can ask for access to a system resource; one column for each system resource that these people can ask for. The number of boxes in this matrix is the product of the number of people and the number of resources. If you double the size of the company, then you double the number of people and the number of resources. This quadruples the number of boxes. If there is a fixed minimum cost to maintaining a check in each box, then the cost of maintaining the matrix grows faster than linear with company growth. Any cost that scales faster than linear is in and of itself a barrier to growth. Security cannot be a barrier to growth, or people will inevitably work around it.

1.173. Discuss the differences between symmetric key cryptography and public key cryptography. Give an example of when you would use each.

Symmetric key cryptography, also called shared key cryptography, involves two people using the same key to encrypt and decrypt information. Public key cryptography makes use of two different keys: a public key for encryption and a private key for decryption. Symmetric key cryptography has the advantage that it is much faster than public key cryptography. It is also generally easier to implement, less likely to involve patented

algorithms, and usually requires less processing power. On the down side, the two parties sending messages must agree on the same private key before securely transmitting information. This is often inconvenient or even impossible. If the two parties are geographically separated, a secure means of communication is needed for one to tell the other what the key will be. In a pure symmetric key scenario, secure communication is generally not available. If it were, there would be little need for encryption to create another secure channel.

Public key cryptography has the advantage that the public key, used for encryption, does not need to be kept secret for encrypted messages to remain secure. This means public keys can be transmitted over insecure channels. Often, people use public key cryptography to establish a shared session key and then communicate via symmetric key cryptography using the shared session key. This solution provides the convenience of public key cryptography with the performance of shared key cryptography.

Both public key and symmetric key cryptography are used to get secure information from the Web. First, your browser establishes a shared session key with the Web site using public key cryptography. Then you communicate with the Web site using symmetric key cryptography to actually obtain the private information.

1.174. If you discover a new cryptography algorithm, should you use it immediately?

This is not a trick question, but goes to the heart of modern cryptography. Basically, no algorithm stays secret for long, and almost every algorithm has at least minor bugs in it or in its implementation. It is virtually impossible to hide an algorithm given the number of people who develop it and know about it and the advanced techniques used by today's best crackers. If your security is based on the secrecy of your algorithm, you have what is called "security by obscurity," which is effectively no security at all. It is very likely that a determined cracker could discover your algorithm. This can render your security worse than useless because you think you have security when, in fact, you have none.

Thus, it is best to make any algorithm public from the beginning and flush

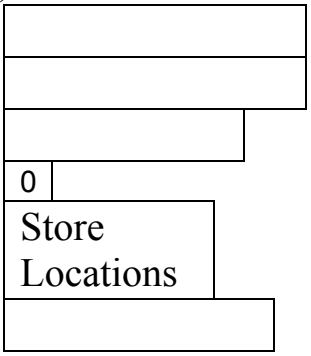
out the bugs rather than keep it secret and have lots of security problems when it is discovered. Only keys should be kept secret. If, after extensive public review and discussion, your algorithm is accepted as secure, then you can probably securely use the algorithm.

1.175. Since FTP is insecure, how can you implement a secure solution that is as easy to use as FTP?

FTP is insecure because it transfers usernames and password as clear text across the network, so any point in the network, which is between you and the FTP server could easily sniff (view) your FTP login information. One way to add a small level of security to FTP is to restrict your server to only allow logins from specific IP addresses (specific_user@specific_host), but the login and password remain as clear text.

The next step in securing file transfers is to use SFTP - Secure FTP. SFTP encrypts all traffic between the client and the server. Depending on the package you choose to install, SFTP is just as easy to install, configure and administer as any regular offering. There are numerous SFTP servers and clients to choose from including WAC SFTP Server, WinSSHD, SSHD-NT, and OpenSSH.

FTP is an insecure protocol that typically includes a cleartext username/password pair, which anyone collecting network traffic can easily intercept, read, and fraudulently reuse. The security-conscious often replace FTP with Secure CP (SCP), which provides similar functionality (file transfer) within a secure shell (SSH) session encrypted on the wire.



1.176.

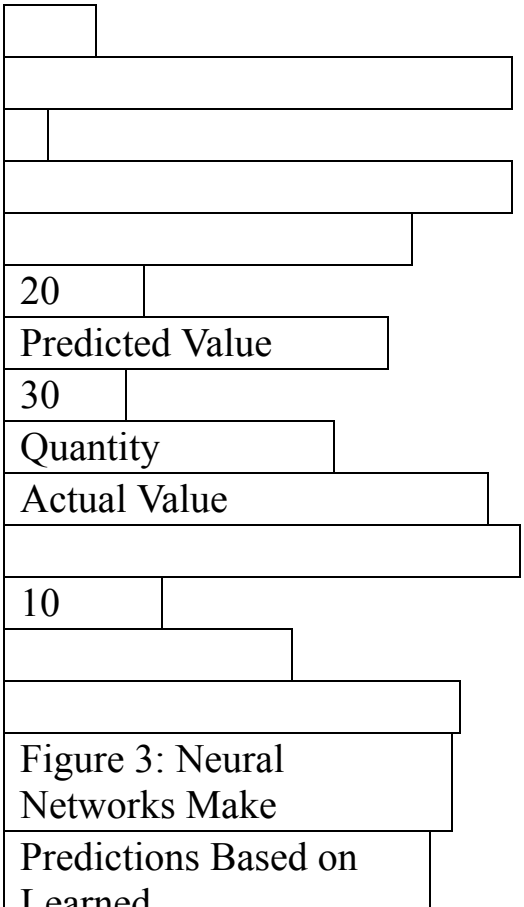
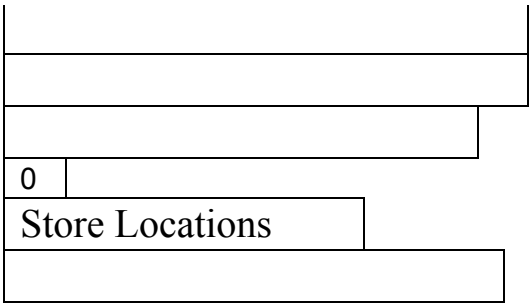


Figure 3: Neural Networks Make Predictions Based on Learned Relationships in Data

Figure 4: Clustering Provides Insight into Attributes of States and the Likelihood of Future Events Occurring



1.177.

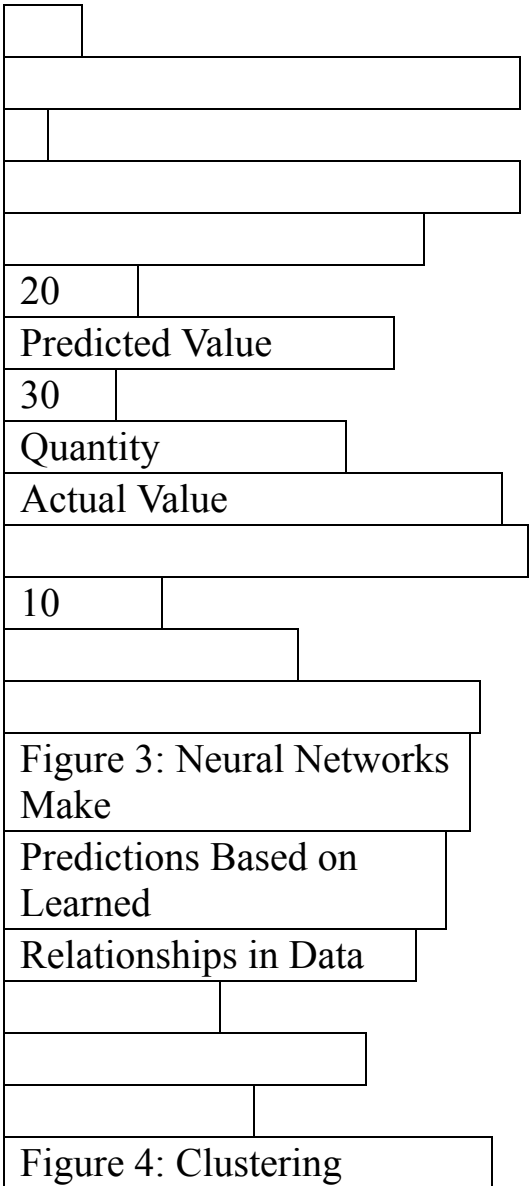


Figure 3: Neural Networks Make Predictions Based on Learned Relationships in Data

Figure 4: Clustering

Provides Insight
into Attributes of States and the
Likelihood of Future Events Occurring
Computer **Associates**

1.178. What is role-based user provisioning?

User provisioning is the process for managing user identity enterprise-wide and beyond. User provisioning encompasses the identification of:

- The types of users an organization will manage
- The systems, applications and other business resources those users will need access to
- The levels of access to those resources users will need
- How the organization will create, update and delete user accounts
- What the strain will be on the IT department to administer the quantity and different types of users
- How the business will guarantee secure access to its resources

User provisioning involves managing a user's life cycle, from creating various user accounts on different systems and extending user access to external services, to temporarily suspending user access or permanently revoking user accounts. Strong user provisioning reduces security risks, including week passwords, and minimizes obstacles to user productivity by increasing access time. User provisioning also provides centralized management capabilities. Using role-based account creation and workflow access rights to business resources, centralized management enables an automated approach across the entire security infrastructure.

1.179. What is HTTPS?

HTTPS is not a single protocol, rather it is the combination of HTTP over an SSL (Secured Socket Layer) transport. Therefore, the properties of both HTTP and SSL are embodied in HTTPS. By including SSL, you gain the following three benefits:

1. encryption: SSL provides encryption using any one of a number of protocols. When SSL establishes a connection, it first negotiates which protocol to use. The result should be the strongest encryption algorithm that both ends support. Of course, if one end supports only a weak encryption algorithm, then the resulting connection will be relatively insecure. For systems where the strength of encryption is important, you should validate that the algorithm chosen is acceptable. You can do this either programmatically or by asking the user, but consider whether the users will all be capable of making an informed determination.
2. identification of parties: If a party involved in setting up an SSL connection has a certificate, then SSL can send this to the other party. This allows the other party to validate the identity of the other first party. This validation is not 100 percent certain, although the mathematical process of the validation is close. The weaknesses arise mainly from human issues, such as a malicious third party having copied the private key of the certificate's legitimate owner.
3. session state: SSL connections are stateful, and because of this, HTTPS can support sessions without the security risks of URL rewriting or cookies.

However, because HTTPS arises directly from SSL, the connection startup is substantially slower. This is a direct result of the negotiation of protocols and cannot be avoided.

HTTP 1.1 keep alive
SSL SSL 3.0, TLS 1.0
PGP Use PGP to encrypt/decrypt files to be exchanged with technology/business partners. Can also use DES encrypted IPSec VPN for better security.
MD5
Inline encryption/decryption Used Twofish in Web pages/URL.

1.180. What are the key differences between IPSec and SSL?

SSL and IPSec are not equivalent remote-access technologies. Some key differences are:

SSL

- Operates at the application layer.
- Supports IP only
- Requires a Web browser that supports SSL.
- Can limit access to specific applications and servers.
- Remote machines can purge traces of remote sessions to protect unauthorized access.

IPSec

- Operates at the network layer.
- Supports any protocol by wrapping it in IP.
- Requires properly configured software client on remote machine.
- Supports unrestricted network access.
- Compromised remote machine can compromise the network.

1.181. What are passfaces?

Passfaces is a graphical password application that substitutes alphanumeric passwords with images of people. While Passfaces (for Windows) provides a second piece of authentication along with a password, both are based on something you know.

Identity exchange

In a basic model to share identities between companies, known as federated identity, a user's identity credential issued on his corporate network can be used to access services on a partner's network. The exchange of credentials is supported by a number of protocols, including the Security Assertion Markup Language, the Liberty Alliance specification or WS-Federation.

The steps involved are:

1. User authenticates to a Web-based application on Server A within his company seeking data that is housed on a database maintained by a partner.
2. Server A passes the users authentication credentials to Server B on the partner network.
3. Server B validates the credentials with Server A.
4. Server B either validates or rejects user's request for access to database.

1.182. What is data marshaling?

The process of gathering data and transforming it into a standard format before it is transmitted over a network so that the data can transcend network boundaries. In order for an object to be moved around a network, it must be converted into a data stream that corresponds with the packet structure of the network transfer protocol. This conversion is known as data marshaling. Data pieces are collected in a message buffer before they are marshaled. When the data is transmitted, the receiving computer converts the marshaled data back into an object.

Data marshaling is required when passing the output parameters of a program written in one language as input to a program written in another language.

To prepare data for processing or for transport over a network. Data marshaling may collect data from a single source or from multiple sources, but then performs the necessary conversion to a common format for processing or for transmission.

1.183. Discuss concerns and limitations while developing for mobile devices.

Limited resources, battery life, unpredictable environment and unreliable communication.

1.184. What is distinctive about real-time software?

Fault tolerance and high reliability, determinance and task scheduling.

1.185. What is the difference between Google Android and Windows Mobile?

When it comes to operating systems for smartphones, there are only a handful of easily identifiable names. This includes Windows Mobile (WM) from Microsoft, Android from Google, Blackberry OS from RIM, and iOS from Apple. Of the four mentioned above, only the first two can be found on different phones from different manufacturers. The biggest difference between Android and WM is that Android is largely based on Linux while Windows Mobile is based on Microsoft Windows. Although the similarities

might be visible in WM, you really cannot tell with Android. Another key difference is licensing, while Windows Mobile is proprietary software, just like Windows, while Android is an open source and basically free operating system, handset manufacturers can tweak it to suit their hardware more.

When it comes to applications that you can install on the handsets, WM still wins out as it has a lot more applications that you can download and install. The only downside to WM is the lack of a centralized app store, like that of Android and iPhone. A WM user who is looking for an app that does a specific thing is basically left on his own.

A major requirement with smartphones is the ability to sync to corporate email systems in order to send and receive emails. Although Google is well known for its email service, it does not scale up to that offered by Microsoft Exchange. Google therefore adopted Microsoft Exchange to Android in order to be able to sync with Exchange servers. This is also the same path that was taken by Apple for the iPhone earlier.

Android is fairly new and Google is still in a very rapid development process in order to fix certain glitches and to add new functionalities to their operating system. On the other hand, Microsoft has virtually abandoned the aging WM with the introduction of Windows Phone 7 Series. A lot of industry watchers have commented that Microsoft created Windows Phone 7 from scratch rather than improve on Windows Mobile.

In summary:

- Android is based on Linux while Windows Mobile is based on Microsoft Windows
- Android is open source while Windows Mobile is not
- Android has its own app market which current Windows Mobile devices lack
- Android still has fewer applications than Windows Mobile
- Android adapts Microsoft Exchange native to Windows Mobile

1.186. What is the difference between iPhone and Windows Mobile?

Since the appearance of the iPhone, there has been much debate about which one is actually better, the iPhone or Windows Mobile smartphones.

Probably the biggest difference between these two devices is in their general approach. The iPhone focuses on simplicity and letting its users do what they want to do without much fuss. Windows Mobile on the other hand, focuses more on productivity and allowing the user to do more complex things than what you can achieve on the iPhone. Because of this, Windows Mobile phones are much harder to learn and get used to when compared to iPhones.

Just to prove a point, the iPhone uses a single home screen, which is easily recognizable to most users. For Windows Mobile, there are a variety of third party applications that let you install and customize your home screen depending on your needs. It is a bit more complicated but should yield a more personalized device.

One major gripe that a lot of people have with the iPhone is the lack of true multi-tasking capabilities. Despite the introduction of multi-tasking in iOS4, this is a very limited feature and not all applications would be able to take advantage of it. Windows Mobile has had true multi-tasking even before the iPhone came to the market.

When it comes to applications for your phones, Apple has total control over which one can be published and which ones cannot. Some developers even refuse to speak out against Apple in fear that their next application would no longer be approved. Microsoft does not exercise any control over any third party software. As long as you have an installer, you can use the application on your phone.

iPhones do not allow user replacement of certain parts like batteries or memory. You are therefore limited to what the iPhone has. In case you run out of battery, there is no other option but to plug-in to the nearest wall socket. Windows Mobile phone allows easy access to the battery and memory so that you can replace them anytime you want and carrying spares is a good option when on long trips.

In summary:

- The iPhone touts simplicity and ease of use while Windows Mobile banks more on flexibility and customizability

- There is only one home screen for the iPhone while you can have a multitude of other interfaces for Windows Mobile
- The iPhone is not able to multi-task while Windows Mobile phones can
- Apple has tight control over what you can install on your iPhone while Windows Mobile users can install pretty much what they want
- The iPhone does not have replaceable memory or batteries while most Windows Mobile phones do

1.187. What is SyncML (Synchronization Markup Language)?

SyncML is an XML-based leading open industry standard for universal synchronization of remote data and personal information across multiple networks, platforms and devices. Ericsson, IBM, Lotus, Motorola, Matsushita Corporation, Nokia, Openwave, Starfish Software, and Symbian are the sponsoring members of the SyncML initiative, and drive the development of the technology together with Promoter members. The objective of the SyncML technology is to enable synchronization of any networked data with any mobile device and to ensure seamless interoperability between devices.

The SyncML initiative recently consolidated into the Open Mobile Alliance (OMA), contributing their technical work to the OMA technical Working Groups: Device Management Working Group and Data Synchronization Working Group.

SyncML's goal is to have networked data that support synchronization with any mobile device, and mobile devices that support synchronization with any networked data. SyncML is intended to work on transport protocols as diverse as HTTP, WSP (part of WAP) and OBEX and with data formats ranging from personal data (e.g. vCard & vCalendar) to relational data and XML documents.

SyncML is designed for use between mobile devices that are intermittently connected to the network and network services that are continuously available on the network. However, SyncML can also be used for peer-to-peer data synchronization. SyncML is specifically designed to handle cases

where network services and mobile devices store the data in different formats or use different software systems.

To ensure interoperability, SyncML describes how common data formats are represented over the network. SyncML permits the definition of new data formats as needs arise, ensuring extensibility. Operators will be able to offer a common interface to their customers, regardless of the type of mobile device. First implementations of SyncML enable users with a SyncML-enabled device to always have an up-to-date calendar and contacts database.

1.188. What are the requirements for SyncML?

A SyncML-compliant server with a SyncML server agent and a synchronization engine is required to complete the end-to-end system. The typical solution comprises a database server and an application server. The SyncML server is integrated with other network elements, such as authentication, billing and profiling, in an operator's network.

A user normally initiates the synchronization session from the terminal. A data call connection (CSD or GPRS) is then established between the SyncML client and the SyncML server. The data interchange begins when the client has been authenticated. The SyncML server manages the synchronization process, during which the following takes place:

- New data is uploaded to the terminal or the application server
- Deleted data is removed from the terminal or the application server
- Modified data is updated in the terminal or the application server

Unmodified data is not exchanged, thereby saving time and precious bandwidth. When the operation is completed, both the server and the client update their log files to keep them up-to-date for the next synchronization session.

1.189. What is Service Oriented Architecture?

IT departments are managing increasingly complex IT portfolios. Yet as business needs change, these departments must still ensure that their technologies remain aligned with business goals. Failure to do so compromises organizational agility.

The problem for IT departments is typically not insufficient functionality; rather, it is that critical business systems such as customer relationship management (CRM) and enterprise resource planning (ERP) operate in isolation from other critical business systems - despite the fact that business processes often span multiple applications. To obtain an end-to-end view of a complex business process necessitates integration of information and process silos. In the past, this has been accomplished either though time-consuming manual interventions, or through hard-coded solutions that are difficult to maintain.

Service orientation is an approach to organizing distributed IT resources into an integrated solution that breaks down information silos and maximizes business agility. Service orientation modularizes IT resources, creating loosely coupled business processes that integrate information across business systems. Critical to a well-designed service-oriented architecture is producing business process solutions that are relatively free from the constraints of the underlying IT infrastructure, because this enables the greater agility that businesses are seeking.

Service Oriented Architecture (SOA) ultimately enables the delivery of a new generation of dynamic applications (sometimes called composite applications). These applications provide end users with more accurate and comprehensive information and insight into processes, as well as the flexibility to access it in the most suitable form and presentation factor, whether through the Web or through a rich client or mobile device. Dynamic applications are what enable businesses to improve and automate manual tasks, to realize a consistent view of customers and partner relations, and to orchestrate business processes that comply with internal mandates and external regulations. The net result is that these businesses are able to gain the agility necessary for superior marketplace performance.

SOA defined: Service orientation is a means for integrating across diverse systems. Each IT resource, whether an application, system, or trading partner, can be accessed as a service. These capabilities are available through interfaces; complexity arises when service providers differ in their operating system or communication protocols, resulting in inoperability.

Service orientation uses standard protocols and conventional interfaces - usually Web services - to facilitate access to business logic and information among diverse services. Specifically, SOA allows the underlying service capabilities and interfaces to be composed into processes. Each process is itself a service, one that now offers up a new, aggregated capability. Because each new process is exposed through a standardized interface, the underlying implementation of the individual service providers is free to change without impacting how the service is consumed.

1.190. Why Service Oriented Architecture?

Complex, distributed IT resources are a concern for businesses. Too frequently, the existing IT portfolio does not adequately meet specific business needs, is costly to manage and maintain, and is inflexible in the face of business growth and change. The solution, however, is not to rip and replace systems or applications, nor to completely renovate them, but rather to find a way to leverage existing IT investments so that overall organizational goals are effectively supported. Service orientation helps to accomplish these goals by making systems more responsive to business needs, simpler to develop, and easier to maintain and manage. Implementing a solution architecture based upon service orientation helps organizations plan ahead for change, rather than responding reactively.

1.191. Who does SOA?

Strictly speaking, SOA is done by developers and solution architects. However, stakeholders in a service-oriented solution span a range of roles, and it is critical that their interests not only be taken into account but that they actively drive the design of the SOA solution. Starting with those interests, the business analyst is concerned with bringing IT investments more in line with the business strategy. For the developer, this means that the SOA solution must map the sources of business information - systems, staff, and trading partners - into a unified and comprehensive view such that the business analyst has greater insight into the costs and benefits of various investments. The chief technology officer (CTO) of the organization will work with developers to ensure that when designing a solution to meet the needs of the business analyst, the integrity of existing IT systems and applications resources are preserved, even as new capabilities are developed. And the IT manager, concerned with effectively integrating

distributed systems such that management is simplified, will work with the developer to ensure that these goals are also met. Ultimately, the developers and solution architects are concerned with creating dynamic collaborative applications that meet the goals of the various stakeholders. The service orientation approach enables them to do so in a way that meets the needs of the organization as a whole.

1.192. What SOA isn't?

There are numerous misconceptions about what SOA is - that it is a product that can be purchased (it is not; it is a design philosophy that informs how the solution should be built); that the goal is to build a SOA (it is not; SOA is a means to an end); or that SOA requires a complete technological and business process overhaul (it does not; SOA solutions should be incremental and built on current investments). SOA is also often equated with Web services, and the terms used interchangeably. While it is true that SOA is made easier and more pervasive through the broad adoption of Web services-based standards and protocols, the two are distinct. SOA is an approach to designing systems - in effect the architectural drawings or blueprint - that directs how IT resources will be integrated and which services will be exposed for use. In contrast, a Web service is an implementation methodology that uses specific standards and language protocols to execute on a SOA solution.

Before starting a SOA: Before a developer writes a single line of code, it is critical to identify both specific business drivers of the SOA endeavor and the dependencies between the business and the underlying technologies. Neglecting the business context can result in a project in which SOA infrastructure is pursued for its own sake, or where investments are made that do not line up well with the needs and priorities of the business.

Two approaches are commonly pursued for implementing SOA: top-down and bottom-up. Both approaches have possible pitfalls that can prevent success. Many organizations that have attempted to roll out SOA infrastructure through a top-down approach have discovered that when the infrastructure is finally delivered it is out of sync with the needs of the business. Likewise, a bottom-up approach can fail as well, because it can lead to a chaotic implementation of services created without regard to

organizational goals. The “middle-out” approach is a successful hybrid of the two other approaches. Business drivers and strategic vision are first employed to set clear direction and priorities. Based on these, the organization takes multiple iterative steps to build out slices of end-to-end capabilities, with each iteration delivering a new, dynamic application back to the business that is used to create business return. Microsoft has long advocated this “real-world” approach to leveraging service-oriented architectures: The approach is focused on rapid time-to-value, and it delivers business results through iterative, incremental steps that facilitate close alignment of IT resources with changing business conditions.

1.193. What is the SOA life cycle?

The core IT assets of any organization include its data, legacy systems, line-of-business applications, packaged applications, and trading partners. Each of these resources is a service provider responsible for producing numerous highly specific outputs, such as inventories and customer data. Service orientation ties together these disparate and autonomous sources of information, bridging a wide range of operating systems, technologies, and communication protocols. The process by which it does this is an iterative one of creating (“exposing”) new services, aggregating (“composing”) these services into larger composite applications, and making the outputs available for consumption by the business user.

Expose: The expose phase of the SOA approach focuses on which services to create from the underlying applications and data. Service creation can be fine-grained (a single service that maps to a single business process) or coarse-grained (multiple services come together to perform a related set of business functions).
The expose phase is also concerned with how the services are implemented. The functionality of underlying IT resources can be made available natively if they already speak Web services, or can be made available as Web services through the use of an adapter.

Compose: Once services are created, they can be combined into more complex services, applications, or cross-functional business processes. Because services exist independently of one another as well as of the underlying IT infrastructure, they can be combined and reused with

maximum flexibility. And as business processes evolve, business rules and practices can be adjusted without constraint from the limitations of the underlying applications.

Consume: Once a new application or business process has been created, that functionality must be made available for access (consumption) by either other IT systems or by end users. The goal of the consumption process is to deliver new, dynamic applications that enable increased productivity and enhanced insight into business performance. Users can consume the composed service through a number of avenues, including Web portals, rich clients, Office business applications, and mobile devices.

1.194. What are the benefits of SOA?

Service-oriented architecture is, first and foremost, a means of attaining greater business agility from existing IT investments. SOA-based solutions connect systems and thereby automate previously manual information-transfer processes whether the goal is to develop new applications; to connect systems, workgroups, or geographically distributed subsidiaries; or to collaborate with trading partners. At the same time, SOA solutions build in the essential services required to ensure that the appropriate resources are accessed by the appropriate users.

SOA benefits accrue for the organization at two different levels, that of the IT organization and that of the business user; in the end, all benefits add up to a dramatic increase in agility and productivity. From the IT department's point of view, SOA-based integration simplifies management of distributed resources across multiple platforms, requires less hardware, is more reliable, is standards-based, and is less costly. From the business point of view, SOA enables development of a new generation of dynamic applications addressing a number of top-level business concerns that are central to growth and competitiveness.

SOA solutions promote:

- Stronger connections with customers and suppliers. By making dynamic applications and business services available to external customers and suppliers, not only is richer collaboration possible, but also customer/partner satisfaction is increased. SOA unlocks critical

supply and demand chain processes - such as outsourcing of specific business tasks - from the constraints of underlying IT architectures, thereby enabling better alignment of processes with organizational strategy.

- Enhanced business decision making. By aggregating access to business services and information into a set of dynamic, composite business applications, decision makers gain more accurate and more comprehensive information. They also gain the flexibility to access that information in the form and presentation factor (Web, rich client, mobile device) that meets their needs.
- Greater employee productivity. By providing streamlined access to systems and information and enabling business process improvement, businesses can drive greater employee productivity. Employees can focus their energies on addressing the important, value-added processes and on collaborative, semi-structured activities, rather than having to conform to the limitations and restrictions of the underlying IT systems.

1.195. What are the challenges associated with SOA?

SOA confers obvious business benefits associated with integration and the creation of new services. However, insufficient attention to governance - the management and monitoring of services, their performance and reliability, and especially their security - can cause inefficiencies and disrupt business processes and the end users they support. As business needs evolve, it is critical to have policies in place that help determine how to prioritize new business processes and services under consideration for implementation, who will be responsible for designing those processes, how they should be implemented, and how the success of the new implementations will be measured. This is especially important given the inherent cross-functional nature of SOA solutions.

Reuse of services, once touted as a primary SOA advantage, is really a byproduct of the approach rather than the goal itself. Reuse is also proving to be more challenging than expected. An existing service may not provide exactly what a different business process requires and so may call for additional work. And designing a service so that it can be reused in the

future requires accurately predicting what future needs will be, something notoriously difficult to do.

1.196. How can your organization get started with SOA?

Make sure that you have sound business drivers. When an organization struggles to justify their SOA projects, it is almost always because they are trying to "do SOA" rather than address a business need. Top-down approaches do not work in the real world. Bottom-up approaches are not manageable either. In contrast, organizations that are successful with SOA often adopt a middle-out approach. These organizations all have something in common - they start with clear business challenges and focus on creating business value.

Try to avoid subscribing to the "build it and they will come" approach. Some organizations spend 18 to 30 months building a services infrastructure. When they finally reach the service consumption or user-experience layer, they find that the business needs have changed, rendering the investments a waste of time and money. It is often more practical to partition your usage scenarios into small sets and build out the entire scenario top to bottom, from the data through to the application consuming the services. Partitioning functionality in this manner can help you track changing business needs much more effectively.

Demonstrate value in rapid iterations. Time-to-value is a critical, healthy metric. The "trust-me" approach is not a healthy model for successfully leveraging SOA. Last, but not least, organizations that have successfully adopted a SOA solution often use a "snowball" approach. How do you build a big snowball? You start with a small snowball. This is probably the most important take-away with respect to leveraging SOA to drive business value.

1.197. What are the benefits of a SaaS solution?

SaaS solution provider can provide software that is:

- Always Available – SaaS solutions that are available from any Internet-connected computer or device—anytime, anywhere.
- Economically Priced – A subscription-based model results in less start-up costs because there is no heavy up-front implementation.

Because SaaS solution provider manages the IT infrastructure, clients benefit from lower IT costs for hardware, software and the people who manage it.

- Painless to Upgrade – SaaS solution provider manages all software improvements, often integrating client feedback, eliminating the need for clients to download or install patches.
- Flexible and Easily Scalable – As the client's user base grows, the SaaS solution can easily grow with them, eliminating the need for purchasing additional hardware, software or bandwidth.

1.198. Give an example of software as a service.

Workflow Software as a Service (SaaS) is a model of software deployment where the workflow server is hosted at the workflow vendor and the workflow functionality is provided to customers across the internet as a rented service. The workflow server and the workflow software are run from SaaS vendor's web-farm and enable customers to subscribe to the workflow software service without the need to buy hardware, software or pay for costly maintenance.

Under the traditional SaaS model, an application resides at an offsite data center where the service provider maintains the data, servers and related hardware. End-users access the application remotely via an Internet browser or back-end web servers. The SaaS model is based on a “one-to-many” delivery model whereby an application is shared across clients, providing a minimal level of application customization to avoid major implementation and integration costs. A key premise of this model is that the SaaS provider invests in the technology, hardware and ongoing support services instead of the client. In return, the customer “pays-as-they-go” according to a subscription and Service Level Agreement (SLA) that ensures the customer a specified level of performance and availability. This approach shares risk between the SaaS provider and customer and offers single-source accountability.

Traditionally, the user purchases a workflow software package and license by paying a one-time fee. The workflow software then becomes the property of the customer who made the purchase. Workflow software as a service, on the other hand, does not have licenses. Rather than a single fee,

payment for the use of the workflow software is through subscription. The user's access and use of the workflow software ends when he stops paying subscription fees.

Workflow SaaS is just part of a wider move towards Internet-based automated services. The infrastructure of the Internet enables to cut out much of the location-dependencies that obstruct business. Workflow software used to be installed on each customer computer and had to be installed in the same building as the people that used it. The web removes these constraints.

1.199. What are the SaaS advantages for the customer?

The SaaS advantages for the customer are:

- Lower Costs. The customer does not need to pay a large upfront licensing fee. They only need to pay recurring subscription fees.
- Less Storage and Backups. The customer does not need to store software or data stored on their computers, so they do not need large data storage facilities. There is also the convenience of not needing to constantly backup data, as storage is SaaS vendor's responsibility.
- Fewer Personnel. Workflow software as a service reduces the need for specially trained IT staff to handle maintenance, monitoring and software updates. SaaS vendor provide these services.
- Savings. The customer saves on hardware, software purchase and updates, support staff, power, security devices, and administration.

The workflow SaaS model is quite different from the licensing model in that large upfront fees, professional services and maintenance costs are replaced by periodic ‘all-in-one’ subscription payments on a monthly or other recurring basis. The tradeoff being that SaaS providers have significantly less upfront revenue in exchange for longer-term and more predictable cash flows as a result of a service-based relationship. Many providers find it easier to build a steady revenue stream this way than with traditional software licensing. Business Process Management (BPM) Software is now being implemented using the SaaS model where a vendor hosts the workflow engine, the process logic and data in a central location and provides end users access to this data and the software which is run and used over the world-wide web.

1.200. Cloud Computing.

Turning the data center into a utility service that provides virtual computing and storage services. You buy processing and storage as you need it. The hardware is of no concern; it appears when you need it.

1.201. Software as a Service.

Delivering applications as a subscription service over the Internet. The ISV that developed the SaaS application runs it, buying deployment infrastructure as a service from utility providers.

1.202. Platform as a Service.

PaaS offers an integrated environment to design, develop, test, deploy and support custom applications. Following the pay-as-you-go model of SaaS, PaaS does not need large up-front investments, and so is a good choice for ISVs.

1.203. Core Cloud Services.

Common features such as billing, security and storage all ISVs need to complete their offer. In the past ISVs would have to build such common features within their on-premise applications. PaaS providers bundle them into a complete offer so ISVs do not have to worry about them.

1.204. Does Cloud Computing offer an easy stepping stone for SaaS?

About two decades ago applications were often delivered on a stack of 3.5" floppy disks that needed to be inserted one disk at a time during the installation process. Over time, the software industry delivery model evolved to CDs and downloading the software over the Internet. Today, many software companies are facing another transition in their delivery model. The Software-as-a-services (SaaS) delivery model, where the software is accessed through a browser or thin client and never installed on the user's desktop, is becoming more prevalent across the industry.

SaaS has many advantages over the traditional software delivery model. The recurring revenue stream, simpler maintenance and application updates, and the lower cost of delivery and distribution are especially

attractive for both the application provider and the end users. However, unlike previous transitions which were changes in manufacturing or delivery technology, the transition to SaaS presents dramatic changes to both the migration and the delivery of the software solution.

Migration - Many companies face the challenge of migrating from a desktop or client-server architecture to a multi-tenant SaaS model. In many cases, the transition to SaaS involves a re-write of the entire application, which can be fraught with delays and incompatibility with legacy systems. This can have a dramatic impact on time-to-market, especially when customers are pressing for a SaaS application in the short term with all of the same features they currently enjoy on their desktop or client-server solution.

Delivery - SaaS changes the delivery requirements dramatically. Since the application is only available online, the solution needs to be available 24 hours a day without fail. For many software companies, this means that the IT delivery infrastructure suddenly becomes core to customer satisfaction. With SaaS hosting, the bar is raised from servers being mostly available, to always available and always online. Many SaaS companies consider colocation or managed server hosting so they can outsource the infrastructure expertise and focus on what they do best - software and application delivery.

Migration to SaaS with private cloud computing - A number of companies have moved their client-server application to a private cloud computing platform as the first step in migrating to a SaaS delivery model. They leverage the benefits of a private cloud and virtual servers to deliver their solution over the Internet. The end user does not know if they're running on in a multi-tenant software application or on a virtual server dedicated to their instance of the software. They see the same set of features they had with the legacy system.

As opposed to the public cloud, the private cloud can deliver a secure platform where the end user can be assured that their data is safe and the network is secure. Each instance of the application can be spun up on its own virtual server within the private cloud. Every time a new customer is

added, an additional virtual server is spun up, with the operating system, application and configuration preloaded and ready to go within minutes.

The private cloud can simplify the process of migrating client-server applications to SaaS. Once the hardware platform is in place, a new virtual server instance can be spun up for each new customer quickly and for minimal incremental costs. Migrating client-server solutions to the private cloud removes the risk of rewriting code to a multi-tenant application from the critical path and delivers the same end user experience as in the client-server model.

Lower Risk, faster migration and preserving the end user experience - all good reasons to consider private cloud computing in your migration to a SaaS delivery model.

1.205. What are some of the challenges of deploying or implementing Software as a Service (SaaS)?

Following are some of the challenges:

Multi-tenant Deployment

A multi-tenant platform is one that uses common resources and a single instance of both the object code of an application as well as the underlying database to support multiple customers simultaneously. Although the technology stack (servers, switches, bandwidth, data storage, etc.) in a multi-tenant deployment are shared and a single instance of code exists, customers' user experience should be similar to that of a user whose application was dedicated on an individual basis, much like that of an on-premise application. Multi-tenant deployment of an application optimizes the use of a limited set of computing resources across a large number of customers.

Multi-tenant deployment can be seen in current Web 2.0 deployments, in which applications aim to facilitate collaboration and sharing between users. This perceived second generation of web-based communities and hosted services, such as social networking sites, utilize centralized application access, and are therefore able to enhance applications based on

real usage statistics, in many cases easier than the on-premise model. Gathering usage statistics and understanding user behavior is anonymized, so the identity of the data source is kept confidential in both consumer and enterprise-class applications. However, this delivery model poses a formidable challenge to delivering reliable and secure application performance, in an isolated yet logically independent, customer interaction.

Since few standards have been established for multi-tenant application delivery or the operational governance to ensure isolation among customers, questions may surface regarding the suitability of the SaaS model for mission-critical applications, or those applications that collect, process, and store sensitive enterprise data. As a result, some customers still choose an isolated tenancy, or on-premise application, to ensure complete isolation. This is a common problem for most innovations coming to the mainstream, as they are guilty of immaturity until proven otherwise.

Several solutions to the issues associated with multi-tenant deployment exist, namely having separate databases per customer, a shared database but separate schemas (set of database tables), or a shared database and shared schemas. A separate database for each customer is the most traditional approach. Although providing each customer their own database simplifies meeting customers' individual concerns for isolation, server (and other infrastructure) provisioning as well as installing multiple instances of a database can become time consuming and difficult to manage. This approach also tends to lead to higher costs, which makes it less favorable. Like the isolated or on-premise approach, the separate schema approach requires more resources to install, configure, and maintain, but differs in that it offers a moderate degree of logical data isolation and can support a larger number of customers per database server. However, the approach that tends to be most favored under SaaS deployment involves using the same database as well as the same schema to store multiple customers' data. When proper design and implementation is achieved, this approach results in the highest efficiencies and has the lowest server and related infrastructure costs because it allows companies to serve the largest number of customers per database server.

Scalability

An application's ability to service user requests in a graceful manner without giving way to lagging response times, while remaining easy to upgrade is many times referred to its scalability. Typically, scalability is the capability of an application to increase total throughput under an increased load when resources (typically more servers) are added.5 For example, an application might be considered scalable if it could be moved from a limited set of servers to a larger configuration of more robust servers, while taking full advantage of the additional server resources and/or processing power in terms of performance (quicker response times) and handling a larger number of simultaneous customers. Typically, it is easier to have scalability upward rather than downward since developers must often use available resources efficiently when an application is initially coded. Therefore, scaling an application downward may mean
trying to achieve the same results in a more constrained environment.

Given SaaS applications are delivered via the Internet, the major challenges that apply to scalability are performance and load management. This aspect of deployment can be affected by many complex factors, including the design of an application's architecture, database schema, network connectivity, available bandwidth, back office services (such as mail, proxy, and security services), and other server resources. Depending on the size and complexity of an application, it may be able to handle anywhere from a handful to tens of thousands of simultaneous customers. Another contributing factor, the number of simultaneous connections made either by a customer through the Internet or through web services will have a direct impact on an application's performance. Therefore, performance objectives must include two scopes: the speed of a single customer's transaction and the amount of performance degradation related to the increasing number of simultaneous connections.

Load management refers to the method by which simultaneous customer requests are distributed and balanced among multiple servers. Effectively balancing loads across servers ensures that they do not become overloaded and eventually unavailable. Load balancing allows an application to scale out across a group of servers, making it easy to add capacity by adding more replicated servers. It also provides redundancy, giving the site

failover, or backup capabilities, so the application remains available to customers even if one or more servers (sometimes referred to as clustering as further described below) fail or need to be taken down for maintenance.

A primary method to achieve scalability is horizontal scaling. This method involves using hardware (such as load balancers) to distribute service requests across multiple servers. Horizontal scaling occurs by using many machines all of which essentially function together as a one, which is also known as clustering. By dedicating several machines to a common task, application fault tolerance is increased. (Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of some of its components.)9 Horizontal scaling provides a method of scalability that is not hindered by server limitations, such as overloading. The key to successful horizontal scaling is location transparency. Location transparency refers to making all business components or services seem as if they reside within the same server or programming space. If any of the application code depends on knowing what server is running the code, location transparency has not been achieved and horizontal scaling will be difficult. This situation is called location affinity. Location affinity requires code changes to horizontally scale an application from one server to many, which can be costly. Thus, strong consideration should be given to location transparency when designing an application.

Reliability

Informally, reliability is the level of accuracy in which an application provides its intended services, usually dictated by user documentation or application specifications. Reliability is about providing correct results and handling error detection and recovery in order to avoid failures. More formally, the mean time between failures (MTBF), that is, the average length of time the application runs until a failure occurs, defines reliability.

Reliable applications are increasingly critical to customers. Because failure of an application can result in lost data (as well as the possibility of lost business, data for analytics, etc.) and considerable recovery costs, companies are requesting 24 X 7 reliability for SaaS applications in terms of a service level agreement (SLA), commonly known as four 9s or 99.99% availability. An SLA is a formally negotiated agreement between two

parties which records the common understanding about services, priorities, responsibilities, guarantees, etc. For example, an SLA may specify the levels of availability, serviceability, performance, operation, or other attributes of an application.11 The Internet has made information and immediate, error-free access to this information, a norm to which customers have become accustomed, making reliability so much more important. Reliability of an application as a whole depends on the reliability of the individual components. Therefore, due to the fact that some components in a system may be related, a failure in one component can affect the reliability of others.

In order to ensure reliability, code review generally takes place before any major application testing (both user interface and background services) begins. Code review can improve the overall quality of an application by examining source code to find and fix mistakes overlooked in the initial development phases. Testing the application through regression, unit, functional, integration and acceptance testing are the best ways to determine its reliability. Regression testing looks for regression bugs, or errors, which prevent the application from behaving as intended. Regression bugs occur as an unintended consequence of program changes, usually to functions that existed prior to a software release or update. Unit testing is used to validate that individual units of source code, or the smallest testable part of an application, are working in the most efficient and error-free manner possible. Functional testing measures the quality of the business components of the system, or underlying code which is created to execute specific functions. The test determines if each business component responds correctly to all conditions that may be presented through inputting of data, workflow associated with data moving correctly from one business component to the next, and that business events are initiated in the order required to meet the business objectives of an applictaion.13 Integration testing is used to verify functional, performance, and reliability requirements placed on the application as a result of passing data from one system to another through some form of common communication protocol. Integration testing can expose problems with the interfaces among application components before errors occur in live application execution.14 Acceptance testing is the last step in the process and allows customers to ensure that the system meets their business requirements - business logic,

application-specific workflow, and screen behavior as anticipated. It is important to have benchmarks in place to measure application reliability. These benchmarks can be measured through regression, unit, functionality, integration, and acceptance testing. Such benchmarks include identifying: faults through severity ratings, the number of continuous hours an application operates without failure, the mean time between failures (MTBF), and reasonable amount of time it takes for an application to recover smoothly.

Usability

The trend in application development is migrating towards a more dynamic user experience. A user interface is the means by which with an application, generally providing a means for input as well as output. Technology has significantly shifted from traditional, static applications that work independently to Web 2.0 applications, which require the capability to collaborate between multiple users and/or applications. Gartner reports that by 2012, consumer technologies will be fully integrated into all settings, including the office, home, remote office and recreational areas. This consumerization has driven the development of the user interface to increase efficiencies.

Many SaaS providers are leveraging Asynchronous JavaScript and XML (Ajax), a current development technique, to improve overall user experience. Ajax is a method of building interactive applications for the Web that process user requests immediately. Combining several programming tools, Ajax allows content on Web pages to update immediately when a user performs an action, unlike other development approaches which users must wait for a whole new page to load. Screens refresh more quickly because only partial data is being sent back and forth, and only certain widgets are being refreshed. For example, a weather forecasting site could display local conditions on one side of the page without delay after a user types in a zip code. By incorporating Ajax, a user interface becomes more efficient, dynamic, and improves response time. In many implementations it allows users to have a better concept of the screen they are viewing and know where they are in a system because the background can remain visible. Ajax can also help reduce cost in software

delivery as smaller packets of data are sent back and forth, ultimately using less bandwidth and processing resources.

The development of the user interface has also accommodated the mobile and flex workforce, whose third screen often takes over more tasks traditionally done on a user's desktop. The third screen refers to a video screen, particularly the screen on a cell phone or personal handheld device that a person uses almost as often as their television or computer screens. When designing an application, developers need to consider who will be accessing the application and from what device. The third screen specifically relates to SaaS because, as a model that parallels to peoples' software usage being available anywhere at any time, users should have the ability to access from mobile devices.

Data Security

Data security is the means of ensuring that data is guarded from corruption and that access to data is controlled. Thus, data security helps ensure the privacy and protection of sensitive information.

The very nature of SaaS poses security challenges. In order to detect and prevent intrusion, strong encryption, authentication, and auditing must be a part of the application design to restrict access to private and confidential data. Liability for the cost and damages associated with any data breach typically rests with the independent software vendor (ISV) and / or managed services provider, individually or collectively referred to throughout this whitepaper as the SaaS provider.19

Therefore, it is imperative for SaaS providers to make certain the SaaS applications they are providing their customers adhere to security policies outlined in any related SLA. Encrypting sensitive data to ensure privacy in such a public space as the Internet should remain a top priority when designing a SaaS application. To encrypt data, information must be encoded in such a way that only the customer or computer with a key can decode it. Two different types of encryption exist: symmetric-key and public-key. In symmetric-key encryption, each computer uses an undisclosed key or code to encrypt information before it is sent over the Internet to another computer. This requires the two computers that will be communicating to

have the necessary key to decrypt the information passed to it. Public-key encryption uses a combination of a private-key and a public-key. The private-key is known only to the computer encrypting the information while the public-key is given to any computer that requires (and is granted) secure communication with it. A popular implementation of public-key encryption is the Secure Sockets Layer (SSL). SSL is an Internet security protocol used by Internet browsers and servers to transmit sensitive information. SSL has become part of an overall security protocol known as Transport Layer Security (TLS).20 Two examples of TLS are Hypertext Transfer Protocol over Secure Socket Layer (HTTPS) and File Transfer Protocol (FTP), both of which encrypt data before transferring it over a secure connection.

Authentication tools provide the ability to determine the identity of a customer attempting to access an application. A typical method of authentication is a username and password. Authentication is rarely used alone; in fact, it is the basis for authorization and privacy, which can be respectfully defined as the determination of whether access to an application will be granted to a particular customer and the act of keeping information from becoming known to unauthorized customers. Authentication on the administrator and developer side is also imperative. When an administrator or developer needs access to object code they would typically validate their identity through token authentication. Tokens are a device characteristically small enough to be carried in a pocket or purse, often designed to attach to a keychain. The purpose of the token is to generate a unique code through an algorithm every “x” number of seconds. (RSA is the chosen security partner of more than 90 percent of Fortune 500 companies.) The number being displayed at that moment on the secure token is used in conjunction with a personal identification number (pin) and password. Because the token changes every “x” number of seconds and must be used in combination with a pin and password, this approach makes data access hackerresistant. This approach can also protect intellectual property (IP) of a SaaS provider in that source, obfuscated, and object code is better protected from misuse. Besides the transmission and retrieval of data, the storage of data is also a pressing security concern. Outsourcing data storage to a managed services provider (a company that provides delivery and management of network-based services, applications, and equipment to enterprises, residences, or other service providers)24 is seen

by many industry experts as the better of three primary alternatives to ensure data security. Managed services providers are able to focus on proper technical maintenance of servers and the remainder of the infrastructure on which a SaaS application resides, including redundant connectivity to the Internet. In addition, most of the leading managed services providers are required to meet auditing standards such as SysTrust and Statement of Auditing Standards No. 70 (SAS70) (further described in the Auditing section below).

Auditing

Audits consist of two types, namely security and information. The first type, a security audit, is when a third party validates a managed services provider's security profile. This is similar to an accounting firm reviewing a company's books. The second type, an information audit, refers to a subsystem that monitors actions to, from, and within an application. For example, an information audit may keep a record of all of the customers and associated users that log onto an application. Such a record is known as an audit trail.

The need to track and log customer activity and transaction flows becomes all the more important, especially since the ratification of the Sarbanes-Oxley Act of 2002 (SOX). Virtually every transaction needs to be logged and/or have a sub-journal created so that after-the-fact analysis of customer activity can be accomplished.26 Thorough and well-known audits such as SysTrust and SAS 70 – more particularly Type II, measure compliance to high standards of data security, and help identify whether security policies are operating efficiently over a period.

SAS 70 was developed and is carried out by The American Institute of Certified Public Accountants (AICPA), while SysTrust was jointly developed and carried out by (AICPA) in conjunction with the Canadian Institute of Chartered Accountants (CICA). SysTrust uses the framework of the Principles of Trust Services and their Criteria when evaluating companies such as managed services providers. In a SysTrust audit, the auditor evaluates whether a particular managed services provider is reliable when compared against the principles set for security, availability, processing integrity, and confidentiality of sensitive information which may

be set forth in the form of an SLA between the managed services provider and its customers. Using these Principles and Criteria the following are determined:

- Security: whether the system is protected against both physical and logical unauthorized access.
- Availability: whether the system is available for operation and use as committed or agreed.
- Processing Integrity: whether processing is complete, accurate, timely, and authorized.
- Confidentiality: whether business information designated as confidential is protected as committed or agreed.

Tests are performed to determine whether a managed services provider's system controls were operating effectively during a specified time period. This time period is typically no shorter than six months and generally one year or greater. If during the audit period controls are operating effectively, an unqualified attestation report is issued. This report addresses whether a managed services provider has maintained effective controls over its system.

On the other hand, SAS 70, is designed more to evaluate a managed services provider's internal controls. The auditor gives an unbiased opinion as to whether internal processes and procedures are suitably and properly designed, put into operation, and operating effectively.

It should be noted that there are two different types of SAS 70 reports, namely Type I and Type II. Type I assesses whether a managed services provider's internal controls are fairly and completely described, and whether they have been adequately designed to meet their objectives. Contrarily, Type II reports are more rigorous and include the auditor's opinion on how effective the internal controls operated under a defined review period. The review period can be as little as six months, but is typically one year or longer. The substantial difference between Type I and Type II reports is that Type I only lists the controls, where Type II tests the efficacy of these controls to ensure the controls are functioning correctly.

The major difference between the SysTrust and SAS 70 audits is the scope of the audit. While the SAS 70 audit provides a report on the effectiveness and adequacy of internal controls for a managed services provider, the SysTrust Audit provides a report covering an application's reliability.

Ownership of Data

While the SaaS model has a number of benefits that make it compelling, one of the more challenging hurdles facing SaaS providers has been data protection and ownership. The party responsible for safeguarding data may very well not be the same party that owns the data. In order to minimize risk and ensure that data is properly safeguarded, it is imperative that data is stored in a SysTrust and SAS-70 compliant environment, regardless of whether the data center is directly operated by the SaaS provider or through a managed services provider (see Data Security section above).

The need to restore data files may arise in as simple a situation as when a single file is inadvertently deleted by a user from primary disk storage or as catastrophic as an on-site disaster. Therefore, a restoration procedure and corresponding disaster recovery plan should be in place, which document specific measures to be taken in the event that data needs to be restored. These set of procedures and/or plan should identify mission critical data and how often it should be backed up, describe where data is housed, backup procedures, what tests are run to ensure the probability of recovery, and what resources are responsible for data recovery. Additionally, a clear path should be detailed as to how easy a customer can retrieve data, both raw and processed, if a decision is made to leave a particular SaaS provider.

Integration

A common definition of integration is the combination of parts so they may work together to form a whole. In technology integration, it many times refers to the act of bringing different applications together to run smoothly as one.

Integrating SaaS applications with back-office systems is an important consideration in achieving the highest level of automation possible. Many applications integrate data or functionality from other applications through the use of mashups, whereby widgets (smaller subsets of data/functionality)

are incorporated into a single viewable screen. The ability to successfully build such interfaces using a variety of protocols, formats, and other methods can be a critical factor when customers evaluate SaaS offerings. Service Oriented Architecture (SOA) is the approach of choice when it comes to many integration strategies.

SOA is an architectural style for creating and using business processes, packaged as services, throughout their lifecycle. SOA also defines and provisions the IT infrastructure to allow different applications, both SaaS and on-premise, to exchange data and collectively participate in business processes. These functions are loosely coupled with the operating systems and programming languages underlying the applications. SOA separates functions into distinct units, or services, which can be distributed over a network and combined to create larger business processes and/or workflow. These services communicate with each other by passing data from one application to another, or by coordinating an activity between two or more services, such as web services with varying degrees of encryption or security (further described under Data Security). SOA allows applications to pass data back and forth without in-depth knowledge of each other's
IT systems behind a firewall. From a business perspective SOA is beneficial because it allows for a more effective integration with business partners and supports customer service initiatives while protecting intellectual property.

Application Updates
Traditionally under the on-premise model the deployment of application updates and patches can be cumbersome and time consuming. Updates are meant to deliver enhancements to an application in order to improve functionality, user interface, or efficiency. Patches, on the other hand, are primarily released to fix problems or bugs with an application or its supporting data structure.

With on-premise applications, updates tend to be larger and usually address major system extensions or improvements. Thus, updates to on-premise applications are scarce and usually occur between four to six months. Updates to SaaS applications, on the other hand, can happen as frequently as every four to six weeks. The reason updates are able to occur so often is

because of the ability to roll out the application in a controlled environment. In other words, because a SaaS application is typically delivered through a multi-tenant platform (further explained in the Multi-Tenant Deployment section) only one instance of the application is being updated versus having to run updates on multiple instances of an application in the on-premise model.

More commonly, SaaS applications are developed using an agile development model. Agile development is a method of software collaboration, which allows for small development updates or iterations of an application to occur more frequently throughout the life cycle of an application. Agile development minimizes risk of creating new bugs because of the minimized scope of development. Development during one unit of time is referred to as an iteration, which commonly last from six to eight weeks. Each iteration is usually comprised of planning, requirements analysis, design, coding, testing, and the creation of user documentation. The goal of each iteration is to have an available application release without software bugs being introduced.

With smaller updates happening more frequently there is less probability of unexpected maintenance outages. Updates are also relatively quick and typically scheduled on days and at times where there is the least amount of user activity (i.e. for business applications, late at night on a weekend), therefore, updates are less likely to disrupt access to an application.

Support

Support is provided in various formats, either electronically or through human intervention to assist a customer in solving a problem or improved usage of an application. There are several ways a SaaS provider can assist customers. For example, support staff may offer to show a customer what steps to take via screen sharing over the Internet or perform a task remotely. Remote support is often used in conjunction with phone support.

According to Mural Ventures, top performing SaaS providers receive nearly 50 percent of their sales through customer referrals, therefore, it is imperative for SaaS providers to offer customers the most responsive and effective support tools possible. Typically, support is broken down into two

tiers, namely first and second tiers. First tier support refers to customers contacting a SaaS provider's support staff for assistance, primarily trained on application usage as defined within user documentation. First tier support is the initial point of contact and typically solves 60 to 80 percent of application-related issues. Often necessary to solve customer's more difficult issues, second tier support many times refers to a SaaS providers ability to make available more technical, and in many cases a software engineer, to help assess and resolve an issue. Second tier support staff is usually also responsible for training key members of first tier support staff, and help to provide further guidance on how to most efficiently troubleshoot an issue.

A major challenge in providing support is the ability to manage and prioritize telephone calls that come into a support center. Call volume is ever changing and can be difficult to predict. Some ways to reduce the number of calls and streamline the support process is to provide back-office tools such as an issue tracking or feedback tracking system, which allows for accurate and up to date reporting of issues (other back-office tools include project and implementation tracking, client billing, mass editing, and events management - further described in the second series of this whitepaper entitled "Back-Office Automation").

Feedback tracking systems are commonly used to create, update, and resolve reported customer issues through tickets. A ticket is a record contained within a feedback tracking system that includes information like submitter, application component, severity, along with ongoing progress being made to resolve an issue. Typically, each ticket has a unique reference number, which allows the support staff to quickly locate, add to, and communicate the status of an issue or request, both internally and externally. Urgency is also assigned to a ticket based on the overall importance of that issue. Other details found in tickets may include a time stamp of when the issue was experienced, date of submission, detailed descriptions of the issue, screen captures, attempted resolution, and other relevant information.

Feedback tracking systems are beneficial because they are an effective accurate way for customers to report the issue that they are experiencing at

the moment it is being experienced, which in turn allows for a higher percentage of relevant tickets to be submitted and greater accuracy of each ticket. As data is collected within a feedback tracking system, it is pooled to create an overall knowledge base of information for decision-making, including improvement of an application. A knowledge base can be very powerful in providing a great degree of customer support, in that it gives support staff the ability to pull information from previous tickets to solve current issues. Feedback tracking systems also reduces the number of phone calls that come into a support center, allowing support staff to better prioritize and attend to issues in a timely manner.

1.206. What are the essential questions for SaaS hosting provides?

'Software as a Service' (SaaS) continues to gain momentum among software business executives as a more cost effective and easier to control delivery mechanism. SaaS has the added benefit of a recurring, predictable revenue stream. However, the move to SaaS often leads ambitious software companies finding themselves with two major headaches:

1. How do they continue to deliver excellent service without making major capital investments in essential IT and facilities infrastructure?
2. How do they minimize the ongoing operating expenses and tech personnel costs of being at the leading edge of a very competitive environment?

Below are 7 essential questions any company should ask before selecting a SaaS hosting provider.

1. Is a 100% service guarantee the same thing as an SLA? SaaS hosting providers offer a certain level of functionality to their customers, with the level of service being specified in a Service Level Agreement (SLA). Many SaaS hosting providers will offer a "guaranteed 100%" uptime. However, this doesn't mean they deliver 100% uptime. Instead, the SaaS hosting providers receive a service credit if 100% uptime isn’t delivered. Therefore companies should carefully review the SaaS hosting provider’s uptime statistics and SLAs to make sure it is consistent with the SLA being offered to customers. Also ask for written details on how they deliver their electrical power, network and server SLAs.

2. Do they get the big picture? Want to know a SaaS company's worst hosting provider nightmare? Signing a contract only to discover that their SaaS hosting provider does little more than deliver 'ping, power & pipe'. What's saved in low monthly costs is lost many times over by not being able to expand the infrastructure in order to support rapid growth, or recover quickly from operational problems. Be sure to gain a full, written understanding of the technical and business relationship that will be in place before signing a contract with a SaaS hosting provider.
3. Does the SaaS hosting provider offer 'high availability' managed dedicated servers? Business-class data centers require significant capital investments to provide and maintain space, power, cooling and network functionality in a way that is cost-effective for SaaS companies to take advantage of. At the server level this translates into being able to deliver a range of affordable, high-end managed server solutions that match business requirements. Contrast that approach with one where a SaaS hosting provider assigns an in-house developer the often hurried task of deploying poorly configured servers for testing, staging and even production. That infrastructure quickly becomes a growth bottleneck due to security, reliability or scalability issues.
4. How do they balance real-time maintenance windows with always-on service? If a SaaS hosting provider's hardware stack is designed properly and redundant hardware is used correctly, full production availability can be maintained even during a maintenance window. Real-time maintenance allows new features to roll out quickly since maintenance windows do not have to be scheduled and announced to customers. Developers will appreciate it because they're able to work more normal business hours, but in a real-time maintenance environment. Users like it too because their confidence in the reliability and stability of the entire application solution increases.
5. How exactly will the managed server hosting solutions keep pace with your business growth? A significant challenge for a SaaS hosting provider is to properly size the "store." It is very important to select a hosting provider who has the infrastructure to support future growth. For example, a company begins with dedicated servers and a managed backup server in the same data center. Then, as clients and

data traffic increase, the business model will require that backup data is securely replicated offsite on a daily basis. But if the managed server hosting provider has only one data center, how will this be achieved? Such unplanned limitations can become major roadblocks for expanding SaaS companies.

6. Does the SaaS hosting provider offer professional incident management and escalation procedures? Nothing is more frustrating as a SaaS hosting provider than getting little to no information about a service issue from the hosting provider. The best SaaS hosting providers have a genuine and very visible culture of service that prevents this from happening to clients. Expect to see systems such as port-level network monitoring plus automated trouble ticket notification in the event of a service issue. Any reputable SaaS hosting provider should be following clearly documented guidelines for all aspects of configuration and change management. If necessary, ask to see these documents. The best SaaS hosting providers will also show a willingness to help with application upgrades by involving their network engineers and Sys Admin staff to minimize the impact of any service disruptions.
7. How can a business be sure that their customers' application data is securely and reliably hosted by the SaaS provider? The Internet has given a whole new meaning to the idea of a 'data bank'. For SaaS hosting providers, it's becoming critical that a trusted business partner securely and reliably hosts their customers' data. To some extent, managed dedicated servers hosted in an outsourced data center have now become a trusted data bank on which the SaaS business model depends.

SaaS businesses that address these 7 questions before choosing a SaaS hosting provider will increase their uptime and minimize their frustrations with their server infrastructure. This lets them focus on the highest value-add portion of the SaaS business model - the application and the users' experience.

1.207. What is REST?

REST (representational state transfer) is an approach for getting information content from a Web site by reading a designated Web page that

contains an XML (Extensible Markup Language) file that describes and includes the desired content. For example, REST could be used by an online publisher to make syndicated content available. Periodically, the publisher would prepare and activate a Web page that included content and XML statements that described the content. Subscribers would need only to know the URL (Uniform Resource Locator) for the page where the XML file was located, read it with a Web browser, interpret the content data using the XML information, and reformat and use it appropriately (perhaps in some form of online publication).

As described in a dissertation by Roy Fielding, REST is an "architectural style" that basically exploits the existing technology and protocols of the Web, including HTTP (Hypertext Transfer Protocol) and XML. REST is simpler to use than the well-known SOAP (Simple Object Access Protocol) approach, which requires writing or using a provided server program (to serve data) and a client program (to request data). SOAP, however, offers potentially more capability. For example, a syndicator that wanted to include up-to-date stock prices to subscribing Web sites might need to use SOAP, which allows a greater amount of program interaction between client and server.

REST is consistent with an information publishing approach that a number of Web log sites use to describe some aspects of their site content, called RDF Site Summary (RSS). RSS uses the Resource Description Framework (RDF), a standard way to describe a Web site or other Internet resource.

2. Mobile Development

2.1. What do you think the future holds for mobile devices?

According to Gartner Group, worldwide smartphone sales will reach 468 million units in 2011, a 57.7 percent increase from 2010. Apple, Android and Windows Phone 7 power 63 percent of all smart phones in 2011, and this will rise to 85 percent by 2015. Separately, in the all-important tablet market, which Gartner estimates will reach 69 million devices this in 2011, Gartner predicts Apple, Android, WebOS will collectively power over 92

percent of tablets sold. Gartner estimates sales of tablets will grow to 294 million units by 2015, of which Apple, Android and WebOS are expected to command a collective 89 percent market share.

2.2. What do you think the future of mobile devices is going to look like? What are some other already new things that we are starting to hear about?

In my opinion, the next big thing is cloud-based mobile computing. There's little disputing the fact that the average consumer is demanding more and more from their handsets. This is driving up the average cost of handsets for more computing power to handle the load. On the business end, handset subsidies have to increase to offset these costs to the consumer while at the same time, the service is more and more commoditized (driving down prices). This is a lose-lose for the carriers.

Enter cloud-based mobile devices. Today, most devices have IP connectivity to the internet at data rates more than sufficient for a reasonable screen refresh rate. Instead of processing data directly on the handset, many applications would get a performance bump by being virtualized on a cloud based back-end.

The technology will be similar to remote desktop solutions like VDI. In this scenario, the mobile device would access its "virtual image" on the cloud. All of the heavy lifting would be handled there (in the cloud) and the screen output would be delivered to the device. The device itself would require little more than a good radio and basic processing/memory. Of course it should also have the capability of caching things like games and certain functions to use off-line, but most things are done online anyway.

There are also several security advantages to this approach that would be very attractive to corporate customers as well. When a handset is lost or stolen, all of the important data is actually on the cloud so it is as simple as de-authorizing the previous device and authorizing a new one.

Mobile application development, distribution, and support become simpler by several orders of magnitude, and the list goes on.

On the business side, handset capabilities are less important and commoditization extends to handsets. This should drive down handset costs and help carriers in the long run. All of this enables lower costs to the customer which drives up demand.

I believe the first carrier to embrace and employ the cloud in this way will have a big win.

2.3. How has the iPad fit into the mobile enterprise?

iPads used belongs to employees. Utilization is just for checking emails, taking notes and replacing paper notepads. No real measurements on productivity impact have been implemented so far. Some top managers are starting to use it, so it might change shortly. I view the ipad as just a "reasonable facsimile" of a laptop, and employees have been using them for ages, so i don't see a huge impact on business. However, like any mobile extension of the workplace, security of company info must be taken into consideration--whatever the tool--smart phone or netbook.

2.4. What trends are developing in web applications?

It is a blurring distinction of traditional applications that you install on your desktop and SaaS (software as a service). In the end, many software will move to SaaS or you can call it Cloud Computing. Only those softwares that require direct access to your PC hardware do require to be installed in your local desktop. It has begun already and it is just the beginning. Web apps will tend to be more and more multi-channel apps where they can be accessed not just via a browser but also via an iPad, an iPhone, android, etc... the app itself will take different shape and form depending on how it is delivered.

2.5. What are the top five new technologies that hold the most promise?

1. Mobile/wireless
2. Virtualization
3. Data management
4. Storage
5. Web services

3. Internationalization and Localization

3.1. What is UTC?

UTC (Universal Time Coordinated) is set by the International Time Bureau (Bureau International de l'Heure) and is the same time as Greenwich Mean Time (GMT), which happens to be the local time in London, England (and Greenwich, too!). Eastern Standard Time (EST) in the U.S. is UTC - 5. Pacific Standard Time (PST) is UTC - 8.

3.2. What is internationalization?

Internationalization is the process of designing an application so that it can be adapted to various languages and regions without engineering changes. Sometimes the term internationalization is abbreviated as i18n (I18N), because there are 18 letters between the first "i" and the last "n." An internationalized program has the following characteristics:

1. With the addition of localization data, the same executable can run worldwide.
2. Textual elements, such as status messages and the GUI component labels, are not hardcoded in the program. Instead they are stored outside the source code and retrieved dynamically.
3. Support for new languages does not require recompilation.
4. Culturally-dependent data, such as dates and currencies, appear in formats that conform to the end user's region and language.
5. It can be localized quickly.

3.3. What is localization?

Localization is the process of configuring software to operate within a specific language, cultural, or national context. Sometimes the term localization is abbreviated as l10n (L10N), because there are 10 letters between the first "l" and the last "n."

3.4. What is the difference between Unicode and UTF-8?

The development of Unicode was aimed at creating a new standard for mapping the characters in a great majority of languages that are being used today, along with other characters that are not that essential but might be necessary for creating the text. UTF-8 is only one of the many ways that you can encode the files because there are many ways you can encode the characters inside a file into Unicode.

UTF-8 was developed with compatibility in mind. ASCII was a very prominent standard and people who already had their files in the ASCII standard might hesitate in adopting Unicode because it would break their current systems. UTF-8 eliminated this problem as any file encoded that only has characters in the ASCII character set would result in an identical file, as if it was encoded with ASCII. This allowed people to adopt Unicode without needing to convert their files or even changing their current legacy software that was unaware of the Unicode standard. Any of the other mapping methods for Unicode breaks compatibility with ASCII and would force people to convert their system.

The observance of compatibility to ASCII of UTF-8 produces a side-effect that makes it ideal for word processing where most of the time, all the characters being used are included in the ASCII character set. UTF-8 only uses a byte to represent every code point resulting in a file size that is half to the same file encoded in UT-16 which uses 2 bytes, and a quarter to the same file encoded in UTF-32 which uses 4.

UTF-8 has been adopted in the World Wide Web because it is both space efficient and byte oriented. Web pages are often simple text files that usually do not contain any character that is outside the ASCII character set. Using other encoding methods would only increase the network load without any benefit. Even in email transport systems, UTF-8 is slowly but surely being adopted as a replacement for the older encoding systems that are still being used.

Summary:

1. Unicode is the standard for computers to display and manipulate text while UTF-8 is one of the many mapping methods for Unicode

2. UTF-8 is a mapping method the retains compatibility with the older ASCII
3. UTF-8 is the most space efficient mapping method for Unicode compared to other encoding methods
4. UTF-8 is the most used Unicode standard for the web

3.5. What is the difference between UTF-8 and UTF-16?

UTF stands for Unicode Transformation Format. It is a family of standards for encoding the Unicode character set into its equivalent binary value. UTF was developed so that users have a standardized means of encoding the characters with the minimal amount of space.UTF-8 and UTF 16 are only two of the established standards for encoding. They only differ in how many bytes they use to encode each character. Since both are variable width encoding, they can use up to four bytes to encode the data but when it comes to the minimum, UTF-8 only uses 1 byte (8bits) and UTF-16 uses 2 bytes(16bits). This bears a huge impact on the resulting size of the encoded files. When using ASCII only characters, a UTF-16 encoded file would be roughly twice as big as the same file encoded with UTF-8.

The main advantage of UTF-8 is that it is backwards compatible with ASCII. The ASCII character set is fixed width and only uses one byte. When encoding a file that uses only ASCII characters with UTF-8, the resulting file would be identical to a file encoded with ASCII. This is not possible when using UTF-16 as each character would be two bytes long. Legacy software that is not Unicode aware would be unable to open the UTF-16 file even if it only had ASCII characters.

UTF-8 is byte oriented format and therefore has no problems with byte oriented networks or file. UTF-16, on the other hand, is not byte oriented and needs to establish a byte order in order to work with byte oriented networks. UTF-8 is also better in recovering from errors that corrupt portions of the file or stream as it can still decode the next uncorrupted byte. UTF-16 does the exact same thing if some bytes are corrupted but the problem lies when some bytes are lost. The lost byte can mix up the following byte combinations and the end result would be garbled.

Summary:

1. UTF-8 and UTF-16 are both used for encoding characters
2. UTF-8 uses a byte at the minimum in encoding the characters while UTF-16 uses two
3. A UTF-8 encoded file tends to be smaller than a UTF-16 encoded file
4. UTF-8 is compatible with ASCII while UTF-16 is incompatible with ASCII
5. UTF-8 is byte oriented while UTF-16 is not
6. UTF-8 is better in recovering from errors compared to UTF-16

4. Frameworks and Tools

4.1. What does "open" mean?

The word has different meanings in different contexts. Our commonsense, every day experience teaches us that "open" is a continuous (not binary) construct. A door can be wide open, mostly open, cracked slightly open, or completely closed. So can your eyes, so can windows, etc.

The "open" in "open content" is a similarly continuous construct. In this context, "open" refers to granting of copyright permissions above and beyond those offered by standard copyright law. "Open content," then, is content that is licensed in a manner that provides users with the right to make more kinds of uses than those normally permitted under the law - at no cost to the user.

Put simply, the fewer copyright restrictions are placed on the user of a piece of content, the more open the content is. The primary permissions or usage rights open content is concerned with are expressed in the "4Rs Framework:"

1. Reuse - the right to reuse the content in its unaltered / verbatim form (e.g., make a backup copy of the content)
2. Revise - the right to adapt, adjust, modify, or alter the content itself (e.g., translate the content into another language)

3. Remix - the right to combine the original or revised content with other content to create something new (e.g., incorporate the content into a mashup)
4. Redistribute - the right to share copies of the original content, your revisions, or your remixes with others (e.g., give a copy of the content to a friend)

Content is open to the extent that its license allows users to engage in the 4R activities. Content is less open to the extent that its license places restrictions (e.g., forbidding derivatives or prohibiting commercial use) or requirements (e.g., mandating that derivatives adopt a certain license or demanding attribution to the original author) on a user's ability to engage in the 4R activities.

4.2. What is Object/Relational Mapping?

Hibernate is concerned with helping your application to achieve persistence. So what is persistence? Persistence simply means that we would like our application's data to outlive the applications process. In Java terms, we would like the state of (some of) our objects to live beyond the scope of the JVM so that the same state is available later.

Specifically, Hibernate is concerned with data persistence as it applies to relational databases (RDBMS). In the world of Object-Oriented applications, there is often a discussion about using an object database (ODBMS) as opposed to a RDBMS. We are not going to explore that discussion here. Suffice it to say that RDBMS remain a very popular persistence mechanism and will so for the foreseeable future.

'Object-Relational Impedence Mismatch' (sometimes called the 'paradigm mismatch') is just a fancy way of saying that object models and relational models do not work very well together. RDBMSs represent data in a tabular format (a spreadsheet is a good visualization for those not familiar with RDBMSs), whereas object-oriented languages, such as Java, represent it as an interconnected graph of objects. Loading and storing graphs of objects using a tabular relational database exposes us to 5 mismatch problems:

1. Granularity: Sometimes you will have an object model which has more classes than the number of corresponding tables in the database (we says the object model is more granular than the relational model). Take for example the notion of an Address.
2. Subtypes (inheritance): Inheritance is a natural paradigm in object-oriented programming languages. However, RDBMSs do not define anything similar on the whole (yes some databases do have subtype support but it is completely non-standardized).
3. Identity: A RDBMS defines exactly one notion of 'sameness': the primary key. Java, however, defines both object identity (a==b) and object equality (a.equals(b)).
4. Associations: Associations are represented as unidirectional references in Object Oriented languages whereas RDBMSs use the notion of foreign keys. If you need bidirectional relationships in Java, you must define the association twice. Likewise, you cannot determine the multiplicity of a relationship by looking at the object domain model.
5. Data navigation: The way you access data in Java is fundamentally different than the way you do it in a relational database. In Java, you navigate from one association to another walking the object network. This is not an efficient way of retrieving data from a relational database. You typically want to minimize the number of SQL queries and thus load several entities via JOINs and select the targeted entities before you start walking the object network.

4.3. What is Hibernate?

Hibernate is an object-relational mapping (ORM) library for the Java language, providing a framework for mapping an object-oriented domain model to a traditional relational database. Hibernate solves object-relational impedance mismatch problems by replacing direct persistence-related database accesses with high-level object handling functions.

Hibernate is free software that is distributed under the GNU Lesser General Public License.

Hibernate's primary feature is mapping from Java classes to database tables (and from Java data types to SQL data types). Hibernate also provides data query and retrieval facilities. Hibernate generates the SQL calls and

attempts to relieve the developer from manual result set handling and object conversion and keep the application portable to all supported SQL databases with little performance overhead.

Mapping: Mapping Java classes to database tables is accomplished through the configuration of an XML file or by using Java Annotations. When using an XML file, Hibernate can generate skeletal source code for the persistence classes. This is unnecessary when annotations are used. Hibernate can use the XML file or the annotations to maintain the database schema.

Facilities to arrange one-to-many and many-to-many relationships between classes are provided. In addition to managing associations between objects, Hibernate can also manage reflexive associations where an object has a one-to-many relationship with other instances of its own type.

Hibernate supports the mapping of custom value types. This makes the following scenarios possible:

- Overriding the default SQL type that Hibernate chooses when mapping a column to a property.
- Mapping Java Enum to columns as if they were regular properties.
- Mapping a single property to multiple columns.

Persistence: Hibernate provides transparent persistence for Plain Old Java Objects (POJOs). The only strict requirement for a persistent class is a no-argument constructor, not necessarily public. Proper behavior in some applications also requires special attention to the equals() and hashCode() methods.

Collections of data objects are typically stored in Java collection objects such as Set and List. Java generics, introduced in Java 5, are supported. Hibernate can be configured to lazy load associated collections. Lazy loading is the default as of Hibernate 3.

Related objects can be configured to cascade operations from one to the other. For example, a parent such as an Album object can be configured to cascade its save and/or delete operation to its child Track objects. This can

reduce development time and ensure referential integrity. A dirty checking feature avoids unnecessary database write actions by performing SQL updates only on the modified fields of persistent objects.

Hibernate Query Language (HQL): Hibernate provides an SQL inspired language called Hibernate Query Language (HQL) which allows SQL-like queries to be written against Hibernate's data objects. Criteria Queries are provided as an object-oriented alternative to HQL.

Integration: Hibernate can be used both in standalone Java applications and in Java EE applications using servlets or EJB session beans. It can also be included as a feature in other programming languages. For example, Adobe integrated Hibernate into version 9 of ColdFusion (which runs on J2EE app servers) with an abstraction layer of new functions and syntax added into CFML.

Entities and components: In Hibernate jargon, an entity is a stand-alone object in Hibernate's persistent mechanism which can be manipulated independently of other objects. In contrast, a component is subordinate to other entities and can be manipulated only with respect to other entities. For example, an Album object may represent an entity but the Tracks object associated with the Album objects would represent a component of the Album entity if it is assumed that Tracks can only be saved or retrieved from the database through the Album object. Unlike J2EE, it can switch databases.

Application programming interface: The Hibernate API is provided in the Java package org.hibernate.

org.hibernate.SessionFactory interface: References immutable and threadsafe object creating new Hibernate sessions. Hibernate-based applications are usually designed to make use only of a single instance of the class implementing this interface (often exposed using a singleton design pattern).

org.hibernate.Session interface: Represents a Hibernate session i.e. the main point of the manipulation performed on the database entities. The latter

activities include (among the other things) managing the persistence state (transient, persisted, detached) of the objects, fetching the persisted ones from the database and the management of the transaction demarcation.

A session is intended to last as long as the logical transaction on the database. Due to the latter feature Session implementations are not expected to be threadsafe nor to be used by multiple clients.

Software components: The Hibernate software includes the following components:

- Hibernate Core – the base software for an object-relational mapping solution for Java environments
- Hibernate Annotations – metadata that governs the transformation of data between the object-oriented model and the relational database model according to the JSR 317 Java Persistence API (JPA 2)
- Hibernate EntityManager – together with Hibernate Annotations, a wrapper that implements a JSR 317 Java Persistence API (JPA 2) persistence solution on top of Hibernate Core
- Hibernate Envers – auditing and versioning of persistent classes
- Hibernate Shards – horizontal partitioning for multiple relational databases
- Hibernate Search – an object-relational mapping for the indexing and search library of Apache Lucene
- Hibernate Tools – a set of tools implemented as a suite of Eclipse plugins and Ant tasks included in JBoss Developer Studio
- Hibernate Validator – the reference implementation of JSR 303 Bean Validation
- Hibernate Metamodel Generator – an annotation processor that creates JSR 317 Java Persistence API (JPA 2) static metamodel classes using the JSR 269 Pluggable Annotation Processing API
- NHibernate – an object-relational mapping solution for the .NET Framework

4.4. What are Hibernate Shards?

You can't always put all your relational data in a single relational database. Sometimes you simply have too much data. Sometimes you have distributed deployment architecture. Sometimes the lawyers say "no".

Whatever your reasons, talking to multiple relational databases inevitably complicates the development of your application. Hibernate Shards is a framework that is designed to encapsulate and minimize this complexity by adding support for horizontal partitioning to Hibernate Core.

Key features:

- Standard Hibernate programming model - Hibernate Shards allows you to continue using the Hibernate APIs you know and love: SessionFactory, Session, Criteria, Query. If you already know how to use Hibernate, you already know how to use Hibernate Shards.
- Flexible sharding strategies - Distribute data across your shards any way you want. Use one of the default strategies provided or plug in your own application-specific logic.
- Support for virtual shards - Think your sharding strategy is never going to change? Think again. Adding new shards and redistributing your data is one of the toughest operational challenges you will face once you've deployed your shard-aware application. Hibernate Sharding supports virtual shards, a feature designed to simplify the process of resharding your data.
- Free/open source - Hibernate Shards is licensed under the LGPL (Lesser GNU Public License)

4.5. Why Hibernate?

For those new to Object/Relational Mapping and/or Java it is recommended to look at the Object/Relational Mapping discussion.

- Natural Programming Model: Hibernate lets you develop persistent classes following natural Object-oriented idioms including inheritance, polymorphism, association, composition, and the Java collections framework.
- Transparent Persistence: Hibernate requires no interfaces or base classes for persistent classes and enables any class or data structure to be persistent. Furthermore, Hibernate enables faster build procedures since it does not introduce build-time source or byte code generation or processing.
- High Performance: Hibernate supports lazy initialization, many fetching strategies, and optimistic locking with automatic versioning and time stamping. Hibernate requires no special database tables or

fields and generates much of the SQL at system initialization time instead of runtime. Hibernate consistently offers superior performance over straight JDBC coding.

- Reliability and Scalability: Hibernate is well known for its excellent stability and quality, proven by the acceptance and use by tens of thousands of Java developers. Hibernate was designed to work in an application server cluster and deliver a highly scalable architecture. Hibernate scales well in any environment: Use it to drive your in-house Intranet that serves hundreds of users or for mission-critical applications that serve hundreds of thousands.
- Extensibility: Hibernate is highly customizable and extensible.
- Comprehensive Query Facilities: Including support for Hibernate Query Language (HQL), Java Persistence Query Language (JPAQL), Criteria queries, and "native SQL" queries; all of which can be scrolled and paginated to suit your exact performance needs.

4.6. What is Ganglia?

Ganglia is a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency. The implementation is robust, has been ported to an extensive set of operating systems and processor architectures, and is currently in use on thousands of clusters around the world. It has been used to link clusters across university campuses and around the world and can scale to handle clusters with 2000 nodes.

Ganglia is a BSD-licensed open-source project that grew out of the University of California, Berkeley Millennium Project which was initially funded in large part by the National Partnership for Advanced Computational Infrastructure (NPACI) and National Science Foundation RI Award EIA-9802069. NPACI is funded by the National Science Foundation and strives to advance science by creating a ubiquitous, continuous, and pervasive national computational infrastructure: the Grid. Current support

comes from Planet Lab: an open platform for developing, deploying, and accessing planetary-scale services.

The ganglia system comprises two unique daemons, a PHP-based web front-end, and a few other small utility programs.

Ganglia Monitoring Daemon (gmond): Gmond is a multi-threaded daemon which runs on each cluster node you want to monitor. Installation does not require having a common NFS filesystem or a database back-end, install special accounts or maintain configuration files.

Gmond has four main responsibilities:

1. Monitor changes in host state.
2. Announce relevant changes.
3. Listen to the state of all other ganglia nodes via a unicast or multicast channel.
4. Answer requests for an XML description of the cluster state.

Each gmond transmits in information in two different ways:

- Unicasting or Multicasting host state in external data representation (XDR) format using UDP messages.
- Sending XML over a TCP connection.

Ganglia Meta Daemon (gmetad): Federation in Ganglia is achieved using a tree of point-to-point connections amongst representative cluster nodes to aggregate the state of multiple clusters. At each node in the tree, a Ganglia Meta Daemon (gmetad) periodically polls a collection of child data sources, parses the collected XML, saves all numeric, volatile metrics to round-robin databases and exports the aggregated XML over a TCP sockets to clients. Data sources may be either gmond daemons, representing specific clusters, or other gmetad daemons, representing sets of clusters. Data sources use source IP addresses for access control and can be specified using multiple IP addresses for failover. The latter capability is natural for aggregating data from clusters since each gmond daemon contains the entire state of its cluster.

Ganglia PHP Web Front-end: The Ganglia web front-end provides a view of the gathered information via real-time dynamic web pages. Most importantly, it displays Ganglia data in a meaningful way for system administrators and computer users. Although the web front-end to ganglia started as a simple HTML view of the XML tree, it has evolved into a system that keeps a colorful history of all collected data.

The Ganglia web front-end caters to system administrators and users. For example, one can view the CPU utilization over the past hour, day, week, month, or year. The web front-end shows similar graphs for memory usage, disk usage, network statistics, number of running processes, and all other Ganglia metrics.

The web front-end depends on the existence of the gmetad which provides it with data from several Ganglia sources. Specifically, the web front-end will open the local port 8651 (by default) and expects to receive a Ganglia XML tree. The web pages themselves are highly dynamic; any change to the Ganglia data appears immediately on the site. This behavior leads to a very responsive site, but requires that the full XML tree be parsed on every page access. Therefore, the Ganglia web front-end should run on a fairly powerful, dedicated machine if it presents a large amount of data.

The Ganglia web front-end is written in the PHP scripting language, and uses graphs generated by gmetad to display history information. It has been tested on many flavours of Unix (primarily Linux) with the Apache webserver and the PHP 4.1 module.

4.7. What is Capistrano?

Capistrano is an open source tool for running scripts on multiple servers; its main use is deploying web applications. It automates the process of making a new version of an application available on one or more web servers, including supporting tasks such as changing databases.

Capistrano is written in the Ruby language and is distributed using the RubyGems distribution channel. It is an outgrowth of the Ruby on Rails web application framework, but has also been used to deploy web applications written using other frameworks, including ones written in PHP.

Capistrano is implemented primarily for use on the bash command line. Users of the Ruby on Rails framework may choose from many Capistrano recipes; e.g. to deploy current changes to the web application or roll back to the previous deployment state.

Capistrano is a utility and framework for executing commands in parallel on multiple remote machines, via SSH. It uses a simple Domain Specific Language borrowed in part from the tool rake. Rake is similar to make in the C world and allows you to define tasks, which may be applied to machines in certain roles. It also supports tunneling connections via some gateway machine to allow operations to be performed behind VPNs and firewalls.

Capistrano was originally designed to simplify and automate deployment of web applications to distributed environments, and originally came bundled with a set of tasks designed for deploying Rails applications. The deployment tasks are now opt-in and require clients to explicitly put "load ‘deploy’" in their recipes.

4.8. What is Apache Lucene?

Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java. It is a technology suitable for nearly any application that requires full-text search, especially cross-platform. Apache Lucene is an open source project available for free download.

Lucene offers powerful features through a simple API:

Scalable, High-Performance Indexing

- over 95GB/hour on modern hardware
- small RAM requirements - only 1MB heap
- incremental indexing as fast as batch indexing
- index size roughly 20-30% the size of text indexed

Powerful, Accurate and Efficient Search Algorithms

- ranked searching - best results returned first
- many powerful query types: phrase queries, wildcard queries, proximity queries, range queries and more

- fielded searching (e.g., title, author, contents)
- date-range searching
- sorting by any field
- multiple-index searching with merged results
- allows simultaneous update and searching

Cross-Platform Solution

- Available as Open Source software under the Apache License which lets you use Lucene in both commercial and Open Source programs
- 100%-pure Java
- Implementations in other programming languages available that are index-compatible

4.9. What is Solr?

Solr is the popular, blazing fast open source enterprise search platform from the Apache Lucene project. Its major features include powerful full-text search, hit highlighting, faceted search, dynamic clustering, database integration, rich document (e.g., Word, PDF) handling, and geospatial search. Solr is highly scalable, providing distributed search and index replication, and it powers the search and navigation features of many of the world's largest internet sites.

Solr is written in Java and runs as a standalone full-text search server within a servlet container such as Tomcat. Solr uses the Lucene Java search library at its core for full-text indexing and search, and has REST-like HTTP/XML and JSON APIs that make it easy to use from virtually any programming language. Solr's powerful external configuration allows it to be tailored to almost any type of application without Java coding, and it has an extensive plugin architecture when more advanced customization is required.

Solr in a Nutshell: Solr is a standalone enterprise search server with a REST-like API. You put documents in it (called "indexing") via XML, JSON or binary over HTTP. You query it via HTTP GET and receive XML, JSON, or binary results.

- Advanced Full-Text Search Capabilities
- Optimized for High Volume Web Traffic
- Standards Based Open Interfaces - XML,JSON and HTTP
- Comprehensive HTML Administration Interfaces

- Server statistics exposed over JMX for monitoring
- Scalability - Efficient Replication to other Solr Search Servers
- Flexible and Adaptable with XML configuration
- Extensible Plugin Architecture

Solr Uses the Lucene Search Library and Extends it!

- A Real Data Schema, with Numeric Types, Dynamic Fields, Unique Keys
- Powerful Extensions to the Lucene Query Language
- Faceted Search and Filtering
- Geospatial Search
- Advanced, Configurable Text Analysis
- Highly Configurable and User Extensible Caching
- Performance Optimizations
- External Configuration via XML
- An Administration Interface
- Monitorable Logging
- Fast Incremental Updates and Index Replication
- Highly Scalable Distributed search with sharded index across multiple hosts
- JSON, XML, CSV/delimited-text, and binary update formats
- Easy ways to pull in data from databases and XML files from local disk and HTTP sources
- Rich Document Parsing and Indexing (PDF, Word, HTML, etc) using Apache Tika
- Apache UIMA integration for configurable metadata extraction
- Multiple search indices

4.10. What is Hudson?

Hudson is a continuous integration (CI) tool written in Java, which runs in a servlet container, such as Apache Tomcat or the GlassFish application server. It supports SCM tools including CVS, Subversion, Git and Clearcase and can execute Apache Ant and Apache Maven based projects, as well as arbitrary shell scripts and Windows batch commands. The primary developer of Hudson was Kohsuke Kawaguchi, who worked for Sun Microsystems at the time. Released under the MIT License, Hudson is free software. The Hudson project is supported by Oracle Corporation.

Builds can be started by various means, including scheduling via a cron-like mechanism, building when other builds have completed, and by requesting a specific build URL.

During recent years Hudson has become a popular alternative to CruiseControl and other open-source build servers. A wealth of plugins have been released for Hudson, extending it far beyond purely being a build tool for Java projects. Plugins are available for integrating Hudson with most version control systems and bug databases. Many build tools are supported via their respective plugins. Plugins can also change the way Hudson looks like or add new functionality. Builds can generate test reports in various formats (JUnit is supported out-of-the-box, others via plugins) and Hudson can display the reports and generate trends and render them in the GUI.

4.11. What is Maven?

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

The following are the key features of Maven in a nutshell:

- Simple project setup that follows best practices - get a new project or module started in seconds
- Consistent usage across all projects means no ramp up time for new developers coming onto a project
- Superior dependency management including automatic updating, dependency closures (also known as transitive dependencies)
- Able to easily work with multiple projects at the same time
- A large and growing repository of libraries and metadata to use out of the box, and arrangements in place with the largest Open Source projects for real-time availability of their latest releases
- Extensible, with the ability to easily write plugins in Java or scripting languages
- Instant access to new features with little or no extra configuration

- Ant tasks for dependency management and deployment outside of Maven
- Model based builds: Maven is able to build any number of projects into predefined output types such as a JAR, WAR, or distribution based on metadata about the project, without the need to do any scripting in most cases.
- Coherent site of project information: Using the same metadata as for the build process, Maven is able to generate a web site or PDF including any documentation you care to add, and adds to that standard reports about the state of development of the project. Examples of this information can be seen at the bottom of the left-hand navigation of this site under the "Project Information" and "Project Reports" submenus.
- Release management and distribution publication: Without much additional configuration, Maven will integrate with your source control system such as CVS and manage the release of a project based on a certain tag. It can also publish this to a distribution location for use by other projects. Maven is able to publish individual outputs such as a JAR, an archive including other dependencies and documentation, or as a source distribution.
- Dependency management: Maven encourages the use of a central repository of JARs and other dependencies. Maven comes with a mechanism that your project's clients can use to download any JARs required for building your project from a central JAR repository much like Perl's CPAN. This allows users of Maven to reuse JARs across projects and encourages communication between projects to ensure that backward compatibility issues are dealt with. We are collaborating with the folks at Ibiblio who have graciously allowed the central repository to live on their servers.

Maven's Objectives: Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal there are several areas of concern that Maven attempts to deal with:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information

- Providing guidelines for best practices development
- Allowing transparent migration to new features

Making the build process easy: While using Maven doesn't eliminate the need to know about the underlying mechanisms, Maven does provide a lot of shielding from the details.

Providing a uniform build system: Maven allows a project to build using its project object model (POM) and a set of plugins that are shared by all projects using Maven, providing a uniform build system. Once you familiarize yourself with how one Maven project builds you automatically know how all Maven projects build saving you immense amounts of time when trying to navigate many projects.

Providing quality project information: Maven provides plenty of useful project information that is in part taken from your POM and in part generated from your project's sources. For example, Maven can provide:

- Change log document created directly from source control
- Cross referenced sources
- Mailing lists
- Dependency list
- Unit test reports including coverage

As Maven improves the information set provided will improve, all of which will be transparent to users of Maven.

Other products can also provide Maven plugins to allow their set of project information alongside some of the standard information given by Maven, all still based on the POM.

Providing guidelines for best practices development: Maven aims to gather current principles for best practices development, and make it easy to guide a project in that direction.

For example, specification, execution, and reporting of unit tests are part of the normal build cycle using Maven. Current unit testing best practices were used as guidelines:

- Keeping your test source code in a separate, but parallel source tree
- Using test case naming conventions to locate and execute tests
- Have test cases setup their environment and don't rely on customizing the build for test preparation.

Maven also aims to assist in project workflow such as release management and issue tracking.

Maven also suggests some guidelines on how to layout your project's directory structure so that once you learn the layout you can easily navigate any other project that uses Maven and the same defaults.

Allowing transparent migration to new features: Maven provides an easy way for Maven clients to update their installations so that they can take advantage of any changes that been made to Maven itself.

Installation of new or updated plugins from third parties or Maven itself has been made trivial for this reason.

What is Maven Not? You may have heard some of the following things about Maven:

- Maven is a site and documentation tool
- Maven extends Ant to let you download dependencies
- Maven is a set of reusable Ant scriptlets

While Maven does these things, as you can read above in the "What is Maven?" section, these are not the only features Maven has, and its objectives are quite different.

Maven does encourage best practices, but we realize that some projects may not fit with these ideals for historical reasons. While Maven is designed to be flexible, to an extent, in these situations and to the needs of different projects, it cannot cater to every situation without making compromises to the integrity of its objectives.

If you decide to use Maven, and have an unusual build structure that you cannot reorganize, you may have to forgo some features or the use of

Maven altogether.

4.12. What is a POM?

A Project Object Model or POM is the fundamental unit of work in Maven. It is an XML file that contains information about the project and configuration details used by Maven to build the project. It contains default values for most projects. Examples for this are the build directory, which is target; the source directory, which is src/main/java; the test source directory, which is src/main/test; and so on.

The POM was renamed from project.xml in Maven 1 to pom.xml in Maven 2. Instead of having a maven.xml file that contains the goals that can be executed, the goals or plugins are now configured in the pom.xml. When executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal.

Some of the configuration that can be specified in the POM are the project dependencies, the plugins or goals that can be executed, the build profiles, and so on. Other information such as the project version, description, developers, mailing lists and such can also be specified.