# Chapter 1 Introduction to Systems Analysis and Design End-of-Chapter Questions

1. **What are the major roles played by a systems analyst on a project team?**

   A system analyst's role is to:

   - Identifying how technology can improve business processes
   - Designing the new business processes
   - Designing the information system
   - Ensuring that the system conforms to information systems standards

2. **Compare and contrast the role of a systems analyst, business analyst, and infrastructure analyst.**

   These three roles emphasize different perspectives on the system. The business analyst represents the sponsor/users interests, while the systems analyst knows how to apply IS to support business needs. Together, the systems analyst and the business analyst can design a system that conforms to the IS standards while adding value to the business. The infrastructure analyst has more technical knowledge and provides the team with technical constraints, or identifies infrastructure changes that the new system will require.

   System Analyst: A System Analysts role is to:

   - Identifying how technology can improve business processes
   - Designing the new business processes
   - Designing the information system
   - Ensuring that the system conforms to information systems standards

   Business Analyst: A Business Analyst's role is to:

   - Analyzing the key business aspects of the system
   - Identifying how the system will provide business value
   - Designing the new business processes and policies

   Infrastructure Analyst: The Infrastructure Analyst's role is to:

   - Ensuring the system conforms to infrastructure standards
   - Identifying infrastructure changes needed to support the system

**3. Compare and contrast phases, steps, techniques and deliverables.**

The Systems Development Life Cycle "SDLC" has a set of four fundamental phases: planning, analysis, design, and implementation. Different projects may emphasize different parts of the SDLC or approach the SDLC phases in different ways, but all projects have elements of these four phases.

Each phase is itself composed of a series of steps, which rely upon techniques that produce deliverables (specific documents and files that provide understanding about the project).

*Example*: When you apply for admission to a university, there are several phases that all students go through: information gathering, applying, and accepting. Each of these phases has steps—information gathering includes steps like searching for schools, requesting information, and reading brochures. Students then use techniques (e.g., Internet searching) that can be applied to steps (e.g., requesting information) to create deliverables (e.g., evaluations of different aspects of universities).

**4. Describe the major phases in the SDLC.**

There are four major phases in SDLC.

a) Planning: The Planning phase is the fundamental process of understanding why an information system should be built and determining how the project team will go about building it.

b) Analysis: The Analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used. During this phase, the project team investigates any current system(s), identifies improvement opportunities, and develops a concept for the new system.

c) Design: The design phase decides how the system will operate, in terms of the hardware, software, and network infrastructure; the user interface, forms and reports; and the specific programs, databases, and files that will be needed.

d) Implementation: The final phase in the SDLC is the implementation phase, during which the system is actually built (or purchased, in the case of a packaged software design). This is the phase that usually gets the most attention, because for most systems it is longest and most expensive single part of the development process.

**5. Which phase in the SDLC is most important? Why?**

All systems development projects follow essentially the same fundamental process called the system development life cycle (SDLC). The planning phase in SDLC is the most important phase because during this phase the project team identifies the business value of the system, conducts a feasibility analysis, and plans the project. This phase is the fundamental process of understanding why an information system should be built and determining how the project team will go about building it.

**6. Describe the principal steps in the planning phase. What are the major deliverables?**

The planning phase is the fundamental process of understanding why an information system should be built and determining how the project team will go about building it. It has two steps:

1. <u>Project Initiation</u>: This step involves the identification of the system's business value. A s*ystem request* presents a brief summary of a business need, and it explains how a system that supports the need will create business value. The IS department works together with the person or department that generated the request (called the *project sponsor*) to conduct a feasibility analysis. The *feasibility analysis* examines key aspects of the proposed project like the technical feasibility, economic feasibility and organizational feasibility

2. <u>Project Management</u>: Once the project is approved, it enters—project management. During project management, the project manager creates a work plan, staffs the project, and puts techniques in place to help the project team control and direct the project through the entire SDLC. The deliverable for project management is a project plan that describes how the project team will go about developing the system.

**7. Describe the principal steps in the analysis phase. What are the major deliverables?**

The analysis phase answers the questions of who will use the system, what the system will do, and where and when it will be used. During this phase, the project team investigates any current system(s), identifies improvement opportunities, and develops a concept for the new system. This phase has three steps:

1. Analysis Strategy: Developed to guide the project team's efforts. Such a strategy usually includes an analysis of the current system (as-is system) and its problems, and then ways to design a new system (to-be system).

2. Requirements Gathering: The analysis of this information—in conjunction with input from project sponsor and many other people—leads to the development of a concept for a new system. The system concept is then used as a basis to develop a set of business *analysis models* that describes how the business will operate if the new system were developed.

3. System Proposal: The analyses, system concept, and models are combined into a document called the system proposal, which is presented to the project sponsor and other key decision makers.

The system proposal is the initial deliverable that describes what business requirements the new system should meet. This is really the first step in the design of the new system.

**8. Describe the principal steps in the design phase. What are the major deliverables?**

The design phase decides how the system will operate, in terms of the hardware, software and network infrastructure; the user interface, forms and reports; and the specific programs, databases, and files that will be needed. The design phase has four steps:

1. Developing design strategy: This clarifies whether the system will be developed by the company's own programmers, whether it will be outsourced to another firm (usually a consulting firm), or whether the company will buy an existing software package.

2. Basic architecture design: This describes the hardware, software, and network infrastructure that will be used. In most cases, the system will add or change the infrastructure that already exists in the organization.

3. Database and file specifications: These define exactly what data will be stored and where they will be stored.

4. Program Design: The analyst team develops the program design, which defines the programs that need to be written and exactly what each program will do.

This collection of deliverables (architecture design, interface design, database and file specifications, and program design) is the *system specification* that is handed to the programming team for implementation. At the end of the design phase, the feasibility analysis and project plan are reexamined and revised, and another decision is made by the project sponsor and approval committee about whether to terminate the project or continue.

**9. Describe the principal steps in the implementation phase. What are the major deliverables?**

The final phase in the SDLC is the implementation phase, during which the system is actually built (or purchased, in the case of a packaged software design). This phase has three steps:

1. System construction: The system is built and tested to ensure it performs as designed. Since the cost of bugs can be immense, testing is one of the most critical steps in implementation.

2. Installation: This is the process by which the old system is turned off and the new one is turned on. It may include a direct cutover approach, a parallel conversion approach, or a phased conversion strategy.

3. Support plan: This plan usually includes a formal or informal post-implementation review, as well as a systematic way for identifying major and minor changes needed for the system.

**10. What are the roles of a project sponsor and the approval committee?**

Project sponsor: Project Sponsor could be represented either by an individual or a department or a team. The project sponsor is from where the request is generated. They are basically involved through out the various phases of the SDLC.

a. The IS department works hand in hand with the project sponsor to conduct a feasibility analysis, be it a technical, organizational or an economical feasibility.

b. During the Analysis phase the project team investigates any current system, identifies improvement opportunities and develops a concept for the new system in conjunction with the input from the project sponsor.

c. At the end of the design phase the feasibility analysis and the project plan are reexamined and revised and the decision whether the project is to continue or terminated is made by the project sponsor along with the approval committee.

d. The key deliverables for each phase are typically very long and are presented to the project sponsor for approval as the project moves from phase to phase.

e. The project sponsor approves the key deliverables for each phase only after which the project moves from phase to phase.

f. The project sponsor along with the users provides valuable comments which are used to re-analyze, re-design and re-implement a second prototype which provides few more features.

Approval Committee: Sometimes referred to as the steering committee is the final decision maker regarding the fate of the project after carefully reviewing the system request, feasibility analysis, system proposal etc.

a) In planning phase after feasibility analysis the approval committee decides whether the project should be undertaken.

b) After Analysis the system proposal is forwarded to the approval committee to decide whether the project should continue to move forward

c) During the Design phase the feasibility analysis and project plan are reexamined and the approval committee will decide whether to terminate or continue with the project

**11. What does *gradual refinement* mean in context of SDLC?**

Generally, the clarity of understanding and the depth of detail of the new system are gradually refined during the phases of the SDLC. Initially, the requirements are only vaguely understood. This understanding is improved during the Analysis phase. Further detail is developed during Design, and then is fully expressed during Implementation.

**12. Compare and contrast process-centered methodologies with data-centered methodologies.**

Process-centered methodologies: Process-centered methodologies emphasize process models as the core of the system concept.
*Example:* Process-centered methodologies would focus first on defining the processes (e.g., assemble sandwich ingredients).

Data-centered methodologies**:** Data-centered methodologies emphasize data models as the core of the system concept.
*Example:* Data centered methodologies would focus first on defining the contents of the storage areas (e.g., refrigerator) and how the contents were organized.

**13. Compare and contrast structured design-based methodologies in general to RAD-based methodologies in general.**

Structured design methodologies are usually fairly formal, step-by-step approaches to systems development. The project moves through the phases in a systematic way. The emphasis in most of these methodologies is development of paper-based specifications for the new system prior to implementation. RAD methodologies, on the other hand, tend to emphasize quick creation of a limited-capability version of the system or a model of the system. These methodologies focus on refining this preliminary system or model rather than trying to fully describe it on paper prior to implementation.

**14. Describe the major elements and issues with waterfall development.**

Waterfall Development follows the phases of the life cycle in sequence (Planning, Analysis, Design, and Implementation). Each phase is thoroughly documented and approval is required before proceeding to the subsequent phase. It is difficult, though not impossible, to go backwards in the SDLC under Waterfall Development.

Waterfall Development requires that the system requirements be precisely specified prior to implementation and also often "freezes" those requirements during development. The high degree of effort devoted to specifying user requirements is a strength of Waterfall Development but specifying those requirements on paper is laborious and may lead to errors and omissions. "Freezing" the requirements during development helps assure that the system is developed according to specifications, but in a dynamic business environment, the system that is ultimately developed may bear little resemblance to what is actually needed at the time the project is completed. Therefore, extensive maintenance may be needed after implementation to revise the system to meet current conditions.

The two key advantages of the structured design waterfall approach are that:

- It identifies system requirements long before programming begins
- It minimizes changes to the requirements as the project proceeds.

The two key disadvantages are that:

a. The design must be completely specified before programming begins.
b. A long time elapses between the completion of the system proposal in the analysis phase and the delivery of the system.

**15. Describe the major elements and issues with parallel development.**

The parallel development methodology attempts to address the problem of long delays between the analysis phase and the delivery of the system. Instead of doing design and implementation in sequence, it performs a general design for the whole system and then divides the project into a series of distinct subprojects that can be designed and implemented in parallel. Once all subprojects are complete, there is a final integration of the separate pieces, and the system is delivered

The major issues in this approach are that it still suffers from problems caused by paper documents. It also adds a new problem such as sometimes the subprojects are not completely independent; design decisions made in one subproject may affect another, and the end of the project may require significant integration efforts.

**16. Describe the major elements and issues with phased development.**

Phased Development is a RAD methodology that does not attempt to develop the complete system initially. Instead, the system is generally specified. User requirements are organized into a series of versions. The first version includes the essential system components and is delivered to the users quickly. Subsequent versions add features and refinements to the system based on the initial specification plus the users' feedback and reaction to using the system.

The critical issue with Phased Development is to accurately specify the initial requirements so that the first version provided to the users is useful, although incomplete. If this is done well, Phased Development will provide value to the organization by getting the users a system to use quickly. New requirements may be identified through user feedback as well, which improves the "fit" of the system to the business needs.

**17. Describe the major elements and issues with prototyping.**

A prototyping-based methodology performs the analysis, design, and implementation phases concurrently, and all three phases are performed repeatedly in a cycle until the system is completed. Prototype is usually the first part of the system that the user will use and the key advantage of a prototyping-based methodology is that it *very* quickly provides a system for the users to interact with, even if it is not ready for widespread organizational use at first.

The major problem with prototyping is that its fast-paced system releases challenge attempts to conduct careful, methodical analysis. Often the prototype undergoes such significant changes that many initial design decisions become poor ones. This can cause problems in the development of complex systems because fundamental issues and problems are not recognized until well into the development process.

*Example:* Imagine building a car and discovering late in the prototyping process that you have to take the whole engine out to change the oil (because no one thought about the need to change the oil until after it had been driven 10,000 miles).

**18. Describe the major elements and issues with throwaway-prototyping.**

In throw-away prototyping, there is a more careful and rigorous analysis performed as compared to Prototyping (more similar to the traditional SDLC). However, in the Design phase, models of various elements of the system are developed to help explore design alternatives and refine system requirements. These prototypes help users clarify their needs, options, and priorities. Once user requirements are established through reaction to the design prototype, it is discarded; however, the requirements it embodied are incorporated into the new system.

Throw-away prototyping is very useful in situations where users are uncertain about key elements of the system. Throw-away prototypes can help focus users on design issues and improve understanding of business needs. The approach helps create a system that suits the users' needs even though those needs may have been poorly understood initially.

**19. What is a use case?**

A use case is a description of a system's behavior as it responds to stimuli from its environment. These stimuli come from the various users of the system as well as from external systems. The use cases are used to identify and to communicate the requirements for the system to the programmers who must write the system.

A use case diagram is a type of behavioral diagram that is part of the Unified Modeling Language (UML) and that graphically depicts a use case or a collection of use cases.

**20. What is meant by use-case driven?**

Use-case driven means that use cases are the primary modeling tools defining the behavior of the system.

**21. Why is it important for an OOSAD approach to be architecture centric?**

A development methodology is said to be architecture centric if the architecture is the main artifact upon which the design and implementation of the system is built. In other words, first the developers decide upon a framework and then the system is built around and upon it. The architecture becomes both an enabler and a constraint on how the system is developed.

It is important of an OOSAD approach to be architecture centric because this approach allows the development team to iterate repeatedly without undermining the stability of the system's core. If the architecture were to be revisited and modified over time, then there would be a very good chance that support to some (or most) system components would eventually be compromised.

**22. What does it mean for an OOSAD approach to be incremental and iterative?**

Iterative and incremental development means that the system undergoes continuous testing and refinement throughout the life of the project. As each increment and iteration is completed, a more complete representation of the user's real functional requirements is uncovered.

**23. Describe the major elements and issues with an object-oriented approach to developing information systems.**

Object-oriented systems analysis and design (OOSAD) is most associated with a phased development RAD-based methodology, where the time spent in each phase is very short. OOSAD uses a use-case-driven, architecture-centric, iterative, and incremental information systems development approach. It supports three different views of the evolving system: functional, static, and dynamic. OOSAD allows the analyst to decompose complex problems into smaller, more manageable components using a commonly accepted set of notations. Additionally, object-oriented systems analysis and design allows the analyst to interact with the user with objects from the user's environment instead of a set of separate processes and data.

**24. Describe the major elements and issues with XP.**

Extreme programming (XP) is founded on four core values: communication, simplicity, feedback, and courage. These four values provide a foundation that XP developers use to create any system. First, the developers must provide rapid feedback to the end users on a continuous basis. Second, XP requires developers to follow the KISS principle.8 Third, developers must make incremental changes to grow the system, and they must not only accept change, they must embrace change. Fourth, developers must have a quality-first mentality. XP also supports team members in developing their own skills. Three of the key principles that XP uses to create successful systems are continuous testing, simple coding performed by pairs of developers, and close interactions with end users to build systems very quickly.

For small projects with highly motivated, cohesive, stable, and experienced teams, XP should work just fine. However, if the project is not small or the teams aren't jelled, then the success of an XP development effort is doubtful. This tends to throw the whole idea of bringing outside contractors into an existing team environment using XP into doubt.

**25. Compare and contrast extreme programming and throwaway prototyping.**

<u>Extreme programming (XP):</u> Extreme Programming is founded on four core values which provide a foundation on which XP developers use to create any system. The four core values are Communication, Simplicity, Feedback and Courage.

<u>Throwaway Prototyping:</u> Throwaway prototyping-based methodologies balance the benefits of well thought out analysis and design phases with the advantages of using prototypes to refine key issues before a system is built. Each of these issues is examined by analyzing, designing, and building a design prototype.

**26. Describe the major elements and issues with SCRUM.**

Scrum development focuses on a few key practices. First, teams are self-organized and self-directed. Unlike other approaches, Scrum teams do not have a designated team leader. Instead, teams organize themselves in a symbiotic manner and set their own goals for each sprint (iteration). Second, once a sprint has begun, Scrum teams do not consider any additional requirements. Any new requirements that are uncovered are placed on a backlog of requirements that still need to be addressed. Third, at the beginning of every workday, a Scrum meeting takes place. Fourth, at the end of each sprint, the team demonstrates the software to the client. Based on the results of the sprint, a new plan is begun for the next sprint.

One of the major criticisms of Scrum, like all agile methodologies, is that it is questionable whether Scrum can scale up to develop very large, mission-critical systems. A typical Scrum team size is no more than seven members.

**27. What are the primary differences between a DevOps methodology and an agile or object-oriented methodology?**

DevOps approaches incorporate ideas from both object-oriented approaches to systems development and lean manufacturing into the agile approaches. The primary extensions that DevOps approaches support are the inclusion of operations personnel with the development team and continuous deployment.

With both object-oriented and agile approaches, deployment was only performed after a version of the system was completed. In other words, the operations personnel took over the system and deployed it. With DevOps approaches, deployment is completed when a new chunk of software has passed a set of relevant tests. Consequently, deployment of a new version of the system is not performed all at once. Instead, the new version is deployed in pieces over time; this is also referred to as a rolling upgrade.

**28. What are the phases and workflows of the Unified Process?**

The phases of the Unified Process are:

1. Inception: a business case is made for the proposed system.
2. Elaboration: requirements are determined and the system is designed.
3. Construction: focuses heavily on programming the evolving information system.
4. Transition: addresses aspects typically associated with the implementation phase of a traditional SDLC approach

The workflows of the Unified Process are:

1. Engineering Workflows
   a. Business Modeling Workflow
   b. Requirements Workflow
   c. Analysis Workflow
   d. Design Workflow
   e. Implementation Workflow
   f. Testing Workflow
   g. Deployment Workflow
2. Supporting Workflows
   a. Project Management Workflow
   b. Configuration and Change Management Workflow
   c. Environment Workflow

**29. Compare the phases of the Unified Process with the phases of the waterfall model.**

The Inception phase of the Unified Process is similar to the Planning phase of the Waterfall methodology.

The Elaboration phase of the Unified Process encompasses most of the activities in both the Analysis and Design phases of the Waterfall methodology.

The Construction and Transition phases of the Unified Process together comprise the set of activities normally performed during the Implementation phase of the Waterfall methodology.

**30. Who is the Object Management Group?**

The Object Management Group is a consortium of organizations whose purpose it is to set standards for object-oriented systems. The OMG is the group that ratified the UML as an industry standard.

**31. What is the Unified Modeling Language?**

The Unified Modeling Language (UML) is a standard set of diagramming techniques. The objective of UML is to provide a common vocabulary of object-oriented terms and diagramming techniques rich enough to model any systems development project from analysis through implementation.

**32. What is the primary purpose of structure diagrams? Give some examples of structure diagrams.**

Structure diagrams provide a way to represent the data and static relationships in an information system. The structure diagrams include class, object, package, deployment, component, and composite structure diagrams.

**33. For what are behavior diagrams used? Give some examples of behavior diagrams.**

Behavior diagrams provide the analyst with a way to depict the dynamic relationships among the instances or objects that represent the business information system. They also allow the modeling of the dynamic behavior of individual objects throughout their lifetime. The behavior diagrams support the analyst in modeling the functional requirements of an evolving information system. The behavior modeling diagrams include activity, sequence, communication, interaction overview, timing, behavior state machine, protocol state machine, and use-case diagrams.

# Chapter 2 Project Management End-of-Chapter Questions

1. **Give three examples of business needs for a system.**

   a) Maintain or improve the competitive position
   b) Perform a business function more efficiently
   c) Take advantage of a new business opportunity

2. **What is the purpose of an approval committee? Who is usually on this committee?**

   The approval committee generally serves as the decision making body regarding business investments in information systems projects. This approval committee could be a company steering committee that meets regularly to make information systems decisions, a senior executive who has control of organizational resources, or any other decision-making body that governs the use of business investments.

   This committee generally has a broad organizational representation, and therefore can avoid allocating resources that will serve only narrow organizational interests. The approval committee commonly has project oversight responsibilities as well as monitoring project performance after the project has been accepted.

3. **Why should the system request be created by a businessperson as opposed to an IS professional?**

   It is important to have a clear understanding of how system will improve business. Companies have now realized that identifying business value and understanding the risks associated with the project are necessary to avoid any potential risks involved. Usually the system originates with a businessperson as he understands the need for the system or system improvement in the business unit and will have a much better idea of the value of the proposed system or improvement and therefore is in a better position to create a meaningful system request

   In fact, the ideal situation is for both IT people (i.e., the experts in systems) and the business people (i.e., the experts in business) to work closely to find ways for technology to support business needs. In this way, organizations can leverage the exciting technologies that are available while ensuring that projects are based upon real business objectives, such as increasing sales, improving customer service, and decreasing operating expenses.

4. **What is the difference between intangible value and tangible value? Give three examples of each.**

   Tangible Value: Tangible value can be quantified and measured easily. Tangible value represents the benefits from the systems that are quantifiable and measurable.

   Example: (a) increased sales (b) reduced operating costs (c) reduced interest costs etc…

   Intangible Value: An intangible value results from an intuitive belief that the system provides important, but hard-to-measure benefits to the organization. Intangible value represents benefits that are real, but are difficult to quantify and measure.

   *Example:* (a) increased customer satisfaction (b) improved decision making (c) better competitive positioning, etc.

5. **What are the purposes of the system request and the feasibility analysis? How are they used in the project selection process?**

   The purpose of the system request is to initiate a systems project. A system request is a document that describes the business reasons for building a system and the value that the system is expected to provide.

   The project sponsor usually completes this form as part of a formal system project selection process within the organization. Most system requests include five elements: project sponsor, business need, business requirements, business value, and special issues

   The feasibility analysis represents a more detailed investigation into the proposed system outlined in the system request. Feasibility analysis guides the organization in determining whether to proceed with a project. Feasibility analysis also identifies the important risks associated with the project that must be addressed if the project is approved.

   The system analyst and the project sponsor work together to more fully develop the objectives of the system and to understand its potential costs and benefits to the organization.

   The system request and the feasibility analysis are the key inputs used by the approval committee in determining if the proposed system has enough merit to move into the Analysis Phase.

6. **Describe two special issues that may be important to list on a system request.**

Environmental factors should be considered (e.g., new governmental reporting requirements); competitive factors (e.g., IS-enabled systems introduced or anticipated by competitors); externally imposed deadlines that cannot be altered (e.g., completion by the start of the next fiscal year); and mandated technologies (e.g., including wireless access).

7. **Describe the three techniques for feasibility analysis.**

Every organization has its own process and format for the feasibility analysis, but most include three techniques:

a. Technical feasibility: Technical feasibility looks at the capability of the organization to successfully develop the proposed system. Included in this assessment is the project size, the types of technologies to be used in the project, and the amount of prior experience with that technology and the business application.

b. Economic feasibility: Economic feasibility addresses the economic justification of the project. Here, we attempt to determine if the value of the project's benefits justifies investing in the project's estimated costs.

c. Organizational feasibility: Organizational feasibility evaluates whether the system is likely to be accepted and used by the organization. Included in this assessment will be the strength of the sponsor's and management's support for the project and the enthusiasm or resistance of the users for the project.

8. **Describe a "risky" project in terms of technical feasibility. Describe a project that would not be considered "risky."**

A project that would be technically risky would be one that is large in scale, utilizes technology that we have little or no experience with, and is for a business area that is new and unfamiliar to the organization.

A project that would not be considered technically risky would be one that is small in scale, uses technology that is well understood, and is for a business area that is very familiar to the users and developers.

**9. What are the steps for assessing economic feasibility? Describe each step.**

The four steps for assessing economic feasibility are:

a. Identify costs and benefits of the proposed system.
b. Assign values to the costs and benefits.
c. Determine the cash flow of the project over the analysis period.
d. Determine the project's investment return.

**10. List two intangible benefits. Describe how these benefits can be quantified.**

a. One example of an intangible benefit is reduced response time to customer requests. Estimating the increase in the number of customers that could be served and the average revenue gained per customer could approximate the value of this benefit. So, if we currently have 1000 customers, the average revenue per customer is $100, and by reducing our response time we can increase the number of customers served by 30%, then our benefit will be $30,000 (300 add'l customers @ $100). (Note: this analysis assumes that demand for the product will increase with our increased capacity. If demand remains constant, however, reduced response time will allow us to save money due to less frequent overtime; allow for greater flexibility in absorbing urgent projects; or create other similar value, though the economic impact of these may be more difficult to estimate).

b. A second example of an intangible benefit is improved customer satisfaction. Determining how much repeat business we lose from dissatisfied customers could approximate the value of this benefit. The amount of repeat business lost could be estimated through customer satisfaction surveys or marketing research. Assume we currently have 1000 customers, each customer brings in average revenue of $100, and we currently lose the repeat business of 10% of our customers due to dissatisfaction. If an improvement in customer satisfaction resulted in losing only 5% of repeat business, then the value of that benefit would be $5,000 (50 customers retained @$100).

**11. List two tangible benefits and two operational costs for a system. How would you determine the values that should be assigned to each item?**

Two tangible benefits are: an increase in sales and a decrease in uncollectible accounts receivable. The best way to measure these benefits is to go to the business people who understand these areas and ask them for reasonable estimates. The sales and marketing managers and the accounts receivable managers will be in the best position to determine these values.

Operational costs are the ongoing costs associated with the new system, and are fairly easy to determine objectively. One common operational cost is that of maintenance agreements for new hardware, which can be determined by contacting hardware vendors about the costs of their maintenance contracts. Another common operations cost is that of new employees that will be needed to run the new system. Salaries and benefits for new employees can be determined by checking local and regional salary and wage surveys for the type of employee needed.

**12. Explain the net present value and return on investment for a cost–benefit analysis. Why would these calculations be used?**

Net Present Value: The net present value (NPV) method compares the present values of the project's cash inflows and outflows. If the present value of the benefits (inflows) is equal to or greater than the present value of the costs (outflows), then the project is considered economically justifiable. NPV has the advantage of including a required rate of return in the calculation, so the NPV figure captures the costs associated with tying up money in the project. NPV also explicitly considers the timing of the cash flows throughout the system life.

Return on Investment: The return on investment (ROI) method simply compares the total net cash flows from the project with the total outflows in aggregate. While this ROI number gives some sense of how much money the project generates in comparison to its total cost, it omits any consideration of the timing of the cash flows and the time value of money. The ROI method, while simple to compute, is flawed in many ways and should not be used as the only economic indicator of a project's merit.

**13. What is the break-even point for the project? How is it calculated?**

Break-even point is defined as the point in time at which the costs of the project equal the value it has delivered. It is determined by looking at the cash flow over time and identifying the year in which the benefits are greater than the costs.

In a project situation, if the project team needs to perform a rigorous cost–benefit analysis, they need to include information about the length of time before the project will break-even, or when the returns will match the amount invested in the project. The greater the time it takes to break-even, the riskier the project.

**14. What is stakeholder analysis? Discuss three stakeholders that would be relevant for most projects.**

Stakeholder analysis is a systematic process that identifies all parties that will be affected by a new information system, and attempts to estimate the consequences of the project for each stakeholder group. A major goal of stakeholder analysis is to ensure that the consequences of a new system are considered for all parties that will be affected by the system.

The most common stakeholders to consider for most systems projects are:

a. The system champion: The system champion is the person or group who initiates the project and provides support for it.
b. The system users: The users are the individuals who will work with the system once it is implemented.
c. The organization's management: The organization management commits resources to the project and has an interest in seeing those resources be used to improve the functioning of the organization.

**15. Why do many projects end up having unreasonable deadlines? How should a project manager react to unreasonable demands?**

Unreasonable deadlines are often the consequence of trying to complete the project to accomplish some business goal rather than being based on a realistic assessment of how long the project will actually take to complete. For example, in the CD Selections case, the project sponsor wants the Internet marketing system to be operational in time to sell CDs for holiday shopping. Too often such external factors are used to create target dates for project completion. The project manager must develop accurate and realistic time estimates for the project, and use these to convince the sponsor that his/her timelines can't be achieved. The project manager is setting the project team up to fail if he/she goes along with a time frame that is known to be unachievable. If the time deadline is immovable, then the project manager should employ time boxing to negotiate a narrowed project scope that will be achievable in the time allotted.

**16. What is the trade-offs that project managers must manage?**

The science (or art) of project management is in making trade-offs among three important concepts:

a) The size of the system (in terms of what it does)
b) The time to complete the project (when the project will be finished)
c) The cost of the project.

A larger project will require more time and money; while a short time frame may require more money or reduced project size. Since most projects have time and/or money constraints, the project manager must strike a balance between size, time, and cost in order to define an achievable project.

**17. Compare and contrast the Gantt chart and the network diagram.**

a) <u>Gantt Chart</u>: In the Gantt chart, horizontal bars are drawn to represent the duration of each task, and as people work on tasks, the appropriate bars are filled in proportionately to how much of the task is finished. Creating a Gantt chart is simple and can be done using a spreadsheet package, graphics software (e.g., Microsoft VISIO), or a project management package.

b) <u>Network Diagram</u>: PERT, which stands for Program Evaluation and Review Technique, is a network analysis technique that can be used when the individual task time estimates are fairly uncertain. Instead of simply putting a point estimate for the duration estimate, PERT uses three time estimates: optimistic, most likely, and a pessimistic. It then combines the three estimates into a single weighted average estimate using the following formula:

$$PERT\ weighted\ average = \frac{Optimistic + (4 \times Most\ likely) + Pessimistic}{6}$$

The PERT chart is drawn as a node and arc type of graph that shows the time estimates in the nodes and the task dependencies on the arcs. Each node represents an individual task, and a line connecting two nodes represents the dependency between two tasks. Pert Chart is the best way to communicate task dependencies because they lay out the tasks as a flowchart in the order in which they need to be completed. The longest path from the project inception to completion is referred to as the critical path.

**18. Some companies hire consulting firms to develop the initial project plans and manage the project, but use their own analysts and programmers to develop the system. Why do you think some companies do this?**

Clearly, the tasks involved in initial project plans and managing the project differ from those involved in the technical development of new systems. Firms may wish to take advantage of the expertise of outside firms in this aspect of development. Other companies do the opposite. They use their own staff to initiate and manage projects but hire consultants, temporary personnel, or outsourcing firms to do the actual programming

**19. What is a use-case point? For what is it used?**

Use-case points is a use cases based project effort estimation approach, which was originally developed based on unique features of use cases and object orientation. From a practical point of view, to estimate effort using use-case points, the use cases and the use-case diagram must have been created.

**20. What process do we use to estimate systems development based on use cases?**

Use case models have two primary constructs: actors and use cases. For use-case point estimation purposes, actors can be classified as simple, average, or complex. Simple actors are separate systems with which the current system must communicate through a well-defined application program interface (API). Average actors are separate systems that interact with the current system using standard communication protocols, such as TCP/IP, FTP, or HTTP, or an external database that can be accessed using standard SQL. Complex actors are typically end users communicating with the system. Once all of the actors have been categorized as being simple, average, or complex, the project manager will count the number of actors in each category and enter the values into the unadjusted actor weighting table contained in the use case point–estimation worksheet. The project manager will then compute the Unadjusted Actor Weight Total (UAW). This is computed by summing the individual results that were computed by multiplying the weighting factor by the number of actors of each type.

A use case represents a major business process that the system will perform that benefits the actor(s) in some manner. Depending on the number of unique transactions that the use case must address, like actors, a use case can be categorized as being simple, average, or complex. A use case is classified as a simple use case if it supports one to three transactions, as an average use case if it supports four to seven transactions, or as a complex use case if it supports more than seven transactions. Once all of the use cases have been successfully categorized, the project manager will enter the number of each type of use case into the unadjusted use case weighting table contained in the use-case point–estimation worksheet. By multiplying by the appropriate weights and summing the results, we get the value for the unadjusted use case weight total (UUCW). Next, the project manager computes the value of the unadjusted use-case points (UUCP) by simply summing the unadjusted actor weight total and the unadjusted use-case weight total.

Use-case point–based estimation also has a set of factors that are used to adjust the use-case point value.

**21. Name two ways to identify the tasks that need to be accomplished over the course of a project.**

The overall objectives for the system should be listed on the system request, and it is the project manager's job to identify all of the tasks that need to be accomplished to meet those objectives.

- One approach for identifying tasks is to get a list of tasks that has already been developed and to modify it. There are standard lists of tasks or methodologies that are available for use as a starting point. (For example: methodology)

- The second approach would be the top to down approach whereby the high-level tasks are defined and then these are broken down into subtasks.

**22. What are the problems associated with conventional WBSs?**

Most approaches to developing conventional WBSs tend to have three underlying problems:

a) *They tend to be focused on the design of the information system being developed.* As such, the creation of the WBS forces the decomposition of the system design and the tasks associated with creating the design of the system prematurely. Where the problem domain is well understood, tying the structure of the work plan to the product to be created makes sense. However, in cases where the problem domain is not well understood, the analyst must commit to the architecture of the system being developed before the requirements of the system are fully understood.

b) *They tend to force too many levels of detail very early on in the SDLC for large projects or they tend to allow too few levels of detail for small projects.* Since the primary purposes of a WBS is to allow cost estimation and scheduling to take place, in conventional approaches to planning, the WBS must be done "correctly and completely" at the beginning of the SDLC. To say the least, this is a very difficult task to accomplish with any degree of validity. These kinds of cases often lead to inaccurate cost and schedule estimation.

c) *Since they are project specific, they are very difficult to compare across projects.* This leads to ineffective learning across the organization. Without some standard approach to create WBSs, it is difficult for project managers to learn from previous projects managed by others. This tends to encourage the "reinventing of wheels" and allows managers to make the same mistakes that previous managers have made.

**23. What is an evolutionary WBS? How do they address the problems associated with conventional WBSs?**

Evolutionary work breakdown structure allows the project manager to provide more realistic estimates for each iteration or build of a system. They also allow the work plan to be decoupled from the architecture of the system, thus allowing projects to be comparable. By supporting comparability among projects, evolutionary WBSs enable organizational learning to take place.

Evolutionary WBSs allow the analyst to address all three of these problems by allowing the development of an iterative work plan:

a) First, they are organized in a standard manner across all projects: by workflows, phases, and then tasks. This decouples the structure of an evolutionary WBS from the structure of the design of the product. This prevents prematurely committing to a specific architecture of a new system.

b) Second, evolutionary WBSs are created in an incremental and iterative manner. The first evolutionary WBS is typically only done for the aspects of the project understood by the analyst. Later on, as the analyst understands more about the evolving development process, more details are added to the WBS. This encourages a more realistic view of both cost and schedule estimation.

c) Third, since the structure of an evolutionary WBS is not tied to any specific project, evolutionary WBSs enable the comparison of the current project to earlier projects. This supports learning from past successes and failures.

**24. What is an iterative workplan?**

Iterative work plans better fit the typical methodologies associated with object-oriented systems development. They allow the project manager to provide more realistic estimates for each iteration or build of a system. Furthermore, they allow the work plan to be decoupled from the architecture of the system, thus allowing projects to be comparable.

**25. What is scope creep, and how can it be managed?**

Scope creep is the most common reason for schedule and cost overruns. The main factor contributing to this is when new requirements are added to the project after the original project scope was defined and frozen. There are other reasons such as: A user may suddenly understand the potential of the new system and realize new functionality that would be useful; a senior manager may decide to let this system support a new strategy that was developed at a recent board meeting etc.,

The keys are to identify the requirements as well as possible in the beginning of the project and to apply analysis techniques effectively. No matter what kinds of precautions are taken, a few factors may go unnoticed or unattended to but several practices can be helpful to control additions to the existing task list.

- The first thing that a project manager needs to do is to allow only absolute necessary requirements to be added after the project begins.
- Any change that is implemented should be carefully tracked so that an audit trail exists to measure the change's impact.
- In some cases, beneficial changes cannot be incorporated into the present system, in which case these additions could be additions to scope as future enhancements to the systems.

**26. What is timeboxing, and why is it used?**

Timeboxing is a technique that is used to organize a project when time is a critical issue. With timeboxing, a fixed deadline is established, and the project team prioritizes the functionality of the system so that the essential features are delivered within the set deadline. If some features must be omitted given that time frame, they are postponed to a later version of the system. With this technique, the users are assured of getting a system with essential functionality by the project deadline, and other, less essential features and refinements are added in later system versions

**27. Create a list of potential risks that could affect the outcome of a project.**

Weak personnel, scope creep, poor design decisions, overly optimistic project estimates

**28. Describe the Kanban method of managing projects.**

The Kanban method uses lots of sticky notes and a large whiteboard that has a set of labeled columns. The number of columns depends on the complexity and size of the project. A small project could simply have columns for story, to do, in progress, testing, and done. For our purposes, a story is similar to a use case. For each story, a sticky note is created and placed in the story column. Also, a sticky note is created for each and every individual requirement necessary to solve the story. Each requirement should be small enough that a team member would be able to complete in a short period of time, e.g., a day or a week. These sticky notes are placed in the To Do column in the same row that the story sticky note was placed. Next, a team member chooses one of the requirements from the To Do column on which they plan to work and places it into the In Progress column. The sticky note remains in the In Progress column until it is either completed or placed into the Testing column or the team member discovers that the requirement is too large. In this case, the team member splits the requirement into a set of sub requirements, creates a sticky note for each sub requirement, and replaces the requirement sticky note with the set of sub requirement sticky notes. The sub requirement sticky notes are then placed in the To Do column. Once a sticky note has been placed into the Testing column, the code that represents the solution to the requirement is tested. If the code passes all of the tests, the sticky note is placed into the Done column. Otherwise, the sticky note is returned to the To Do column. This process is repeated over all stories related to the system until all requirements-based sticky notes are in the Done column.

**29. Describe Tuckman's five stages of small group development.**

The stages include forming, storming, norming, performing, and adjourning.

1. The forming stage deals with the startup characteristics that a new team possess.
2. The storming stage sees members attempting to "flex their muscles" in an attempt to demonstrate their individual worth to the group.
3. In the norming stage, the characteristics of the group moves much more toward becoming a team.
4. The performing stage is the stage that the group becomes a team and not simply a loose connection of individuals working together.
5. The adjourning stage is the stage that the team enters once the project has been completed.

**30. Define the five characteristics of a jelled team.**

A jelled team is a group of people so strongly knit that the whole is greater than the sum of the parts. They have the following five characteristics:

- First, jelled teams have a very low turnover during a project.
- Second, jelled teams have a strong sense of identity.
- Third, the strong sense of identity tends to lead the team into feeling a sense of eliteness.
- Fourth, during the development process, jelled teams feel that the team owns the information system being developed and not any one individual member.
- Fifth, team members really enjoy doing their work.

**31. Describe the differences between a technical lead and a functional lead. How are they similar?**

Technical Lead: The technical lead is typically a project team member who supervises the programmers and more technically oriented project staff.

Functional Lead: The functional lead is a team member who oversees the systems and business analysts on the team. Both positions report to the project manager, and are responsible for managing, controlling, and coordinating the work of their assigned team members

**32. Describe three technical skills and three interpersonal skills that would be very important to have on any project.**

Desirable technical skills might include programming experience in the chosen programming language, experience in configuring the hardware and communications technology platform correctly, and experience in utilizing the file/database environment effectively. Desirable interpersonal skills might include interviewing skills, negotiation skills, and conflict resolution skills.

**33. What are the best ways to motivate a team? What are the worst ways?**

Research has shown that technically oriented people are motivated by recognition, achievement, the work itself, responsibility, advancement, and the chance to learn new skills. The worst ways to motivate technical staff include setting unrealistic deadlines, failing to recognize good effort, accepting low quality output, rewarding all team members monetarily regardless of work quality, failing to include team members in important project decisions, and providing poor working conditions.

**34. List three techniques to reduce conflict.**

Clearly define the roles on the project, hold team members accountable for their assigned tasks, develop detailed operating procedures and make sure the team members understand them, have each team member commit to the project charter.

**35. Describe how to make meetings more effective.**

- First, meeting leadership must become an organizational priority.
- Second, meetings should have an agenda and the fewer the items, the better.
- Third, the length of meetings should be minimized.
- Fourth, only participants that can help reach the goals of the meeting should be invited to participate.
- Fifth, vary the location and seating.

**36. Describe three types of standards, and provide examples of each.**

Coding standards define the content and structures that are to be used in programs. An example would be that all programs are to be written following structured programming guidelines.

Procedural standards define processes that are to be followed by all team members. An example would be required attendance at a weekly team progress meeting, and required honest progress reporting at that meeting.

User interface design standards create a common understanding of the appearance and functioning of the screens the end users see. An example would be to create a standard group of icons that are used consistently on all screens.

**37. What belongs in the project binder? How is the project organized?**

All project deliverables, all internal communication, and all project documentation should be placed in the project binder. The sections of the project binder should follow the phases of the life cycle, and each deliverable produced during the project should be placed in its appropriate place.

# Chapter 3 Requirements Determination
# End-of-Chapter Questions

1. **What are the key deliverables that are created during the analysis phase? What is the final deliverable from the analysis phase, and what does it contain?**

   The Analysis Phase takes the general ideas in the system request and refines them into a detailed requirements definition and answers the questions of *who* will use the system, *what* the system will do, and *where* and *when* it will be used. During this phase, the project team will learn about the system. The team then produces the Functional Model (Activity Diagrams, Use Case Descriptions and Diagram), Structural Model (CRC Cards and Class and Object Diagrams), and Behavioral Models (Sequence Diagrams, Communication Diagrams, and Behavioral state machines) that together form the system proposal.

   The system proposal also includes revised project management deliverables, such as the feasibility analysis and the work plan. The system proposal is presented to the approval committee, who decides if the project is to continue.

2. **What is the difference between an as-is system and a to-be system?**

   An as is system is an existing or a current state of system or process. The As-Is system is the combination of people, processes, data, and technology that currently perform the tasks and functions of the system under study. The As-Is system may or may not incorporate computers.
   A to be system is the one which is developed on the basis of the analysis done on the current system which needs to be changed. The To-Be system is the combination of people, processes, data, and technology that will perform the required tasks and functions of the system under study.

3. **What are the three basic steps of the analysis process? Which step is sometimes skipped or done in a cursory fashion? Why?**

   The basic process of analysis is divided into three steps:

   a) Understanding the as-is system
   b) identifying improvements
   c) Developing requirements for the to-be system.

   Sometimes the first step (i.e., understanding the as-is system) is skipped or done in a cursory manner. This happens when no current system exists, if the existing system and processes are irrelevant to the future system, or if the project team is using a RAD or agile development methodology in which the as-is system is not emphasized.

4. **Compare and contrast problem analysis and root cause analysis. Under what conditions would you use problem analysis? Under what conditions would you use root cause analysis?**

   Problem Analysis and Root-Cause Analysis are two different techniques to be employed in Business Process Automation to determine improvements to the current system. The two strategies vary in the emphasis of the analysis performed.

   Problem Analysis
   Problem Analysis asks the users and managers of the As-Is system to identify system problems and to suggest problem solution. Problem analysis would be suitable when the problems being experienced with the As-Is system are relatively minor, and the changes needed are primarily 'touch-ups'.

   Root-Cause Analysis
   Root-Cause analysis focuses on being sure that the problem's underlying cause is understood, rather than just assuming that cause is known. This emphasis helps ensure that solutions chosen will solve real business problems rather than solving a problem symptom.

   Root-Cause Analysis is appropriate when the problems of the As-Is system are more significant, and the team needs assurance that they are designing a solution that really solves the true problems.

5. **Compare and contrast duration analysis and activity-based costing**

Duration analysis and activity-based costing are techniques used in Business Process Improvement to help identify system improvement opportunities. These two techniques focus on existing business processes in the As-Is system.

"Duration Analysis" assesses the time requirements to complete a process. First, the total time required to complete a business process is determined. Then the process is broken down into individual steps, and the time required to complete each step is determined. The total time of all steps is calculated and compared to the total time of the process. If these totals are substantially different (total time of process > total time of process steps), then significant inefficiencies exist, and the process needs major revision.

"Activity-Based Costing" assesses the costs required to perform a process. In this technique, the cost of each major process or each step in a business function is measured. The most costly processes are then the targets of the development team's improvement efforts. Although straightforward in concept, this technique is complex in practice due to the difficulty of determining the indirect costs to apply to the business process (es). Incorrectly assigned indirect costs may bias the results of the analysis.

6. **Describe the five major steps in conducting interviews.**

The five major steps to conducting interviews are:

a) Selecting interviewees - determine who should be interviewed, why they should be interviewed (what contribution will they make to the project?), and develop a schedule for conducting the interviews.
b) Design the interview questions - depending on who is being interviewed and the type of information desired, the analyst needs to design the interview session with the appropriate structure and question type.
c) Prepare for the interview - review related material; review interview plan; review interview questions and plan for any anticipated problem areas; inform interviewee about interview agenda.
d) Conduct the interview - establish rapport with the interviewee; explain purpose of interview; ask interview questions; record information from interviewee.
e) Prepare post-interview report - summarize the interview in an interview report.

7. **Explain the difference between a closed-ended question, an open-ended question, and a probing question. When would you use each?**

   <u>Closed-ended question</u>
   Closed-ended questions require a specific answer, Closed-ended questions are used to capture specific, factual information.

   <u>Open-ended question</u>
   Open-ended questions leave room for the interviewee to elaborate on the question in their answer. Open-ended questions are used to gather a broader, rich information set. Open-ended questions can help the interviewer learn why things are the way they are, and also give the interviewee the chance to add ideas or issues that the interviewer did not anticipate.

   <u>Probing question</u>
   Probing questions are follow-up questions that ask for more information or examples. Probing questions are used whenever the interviewer is not satisfied with his/her understanding of the interviewee's answer, and needs more explanation before moving on to another topic.

8. **Explain the differences between unstructured interviews and structured interviews. When would you use each approach?**

   Unstructured interviews are interviews that are planned to include broad, far-ranging questions. Often open-ended questions are used to gather information. These interviews are most likely to be used early in the information gathering process, when few details are known, and the analyst is trying to understand the basic business process and the As-Is system.

   Structured interviews are interviews that are planned to gather very specific, detailed information. These interviews use more closed-ended questions that zero in on specific information and facts. These interviews will be conducted later in the information gathering process, when the analyst has learned enough about the business process in order to formulate more specific, detailed questions.

9. **Explain the difference between a top-down and bottom- up interview approach. When would you use each approach?**

These two interview structures vary in the way the interview questions are organized within the interview session. The top-down approach begins with broad, general issues, and moves gradually toward more specific questions. The bottom-up approach is the opposite; beginning with very specific questions and moving to broad, general questions. The top-down approach is most common, because it allows the parties to develop a shared understanding of the general situation before getting to details. Using the bottom-up approach can be difficult and non-productive unless the interviewer has already learned quite a bit about the situation is only needs to verify or elaborate on some items. Also, some interviewees cannot be expected to really provide answers to broad, general questions, in which case the bottom-up approach is appropriate.

10. **How can you differentiate between facts and opinions? Why can both be useful?**

An opinion is a statement about an issue or situation that may or may not be support by fact. If it is stated "Most of our collections are on-time," this is an opinion that can be confirmed or denied by doing an actual measurement of on-time collections. This information is factual, and may provide the basis for the opinion expressed. Alternatively, the opinion may be a misstatement of actual fact, and may suggest an area where there is misunderstanding of the true situation.

11. **How does designing questions for questionnaires differ from designing questions for interviews?**

Because the information on a questionnaire cannot be immediately clarified for a confused respondent, developing good questions is critical for questionnaires. Questions on questionnaires must be very clearly written and leave little room for misunderstanding.

12. **What are typical response rates for questionnaires and how can you improve them?**

Paper and e-mail questionnaires have typical response rates of 30-50%. Web-based questionnaires have lower response rates of 5-30%. To improve response rates, many methods have been devised, such as informing the respondent why s/he was selected and why the questionnaire is being sent; stating a specific return due date; offering an inducement; offering to provide a summary of responses; personally requesting that the questionnaire be completed; following up non-responses; and coercion by management.

**13. What are the key aspects of using observation in the information-gathering process?**

Observation is very helpful in enabling the analysts to understand the As-Is system. It is often much easier to grasp a process by observing it rather than having it explain verbally. Observation helps validate information learned from other techniques. As an observer, one must always bear in mind that people's behavior may change because they are being observed. Therefore, the results of observation may be questionable. The behavior that is observed is not necessarily the true behavior.

**14. What is document analysis?**

Document analysis focuses on existing documentation of the current system, forms and reports that are a part of the current system, plus any personal forms, reports, or files that have been developed informally by the end users. By studying this material the analysts can gain insight into the existing system, how it is used, and possibly also aspects of the system that are not being used.

**15. How does the formal system differ from the informal system? How does document analysis help you understand both?**

The formal system consists of the forms, reports, policies, and procedures that were established when the system was first created. Over time, users often modify their use of the system; as user needs change, the formal system may not change, so users adapt by creating an informal system. The informal system is the actual forms, reports, policies, and procedures that are currently used by the system's users Document analysis entails reviewing the documentation and examining the system itself. It can provide insights into the formal and informal system. Under ideal circumstances, the project team that developed the existing system will have produced documentation, which was then updated by all subsequent projects. In this case, the project team can start by reviewing the documentation and examining the system itself.

**16. Explain factors that can be used to select information-gathering techniques.**

The different information gathering techniques all have strengths and weaknesses, and the astute analyst will use a combination of techniques in any project. The analyst should select the techniques based on the type of information being sought, the breadth and depth of information needed, the degree to information needs to be integrated, the need for user involvement, and the cost of the technique. Interviews and JAD sessions are the most productive information gathering methods, however, these techniques require the most skilled analysts to conduct

**17. Describe the different approaches for text analysis.**

- A very useful, but old tool that can be used to perform some simple text analysis is sentence diagraming. The tool forces the student to “draw” a sentence using a set of lines that represent the different parts of speech.
- Domain analysis concentrates on the meaning or semantics, and possibly pragmatics, of the objects of interest. Domain analysis begins by trying to identify the semantic relationships among the objects.
- Taxonomic analysis builds up a node and arc type of graph by using the semantic relationships to connect the different objects of interest in the problem domain.
- Where taxonomic analysis focuses on the creation of a taxonomy, componential analysis specifically focuses on the attribution semantic relationship.
- Theme analysis attempts to identify any underlying themes that were uncovered during the requirements gathering process.

**18. What is the purpose of the requirements definition?**

The requirements definition report—usually just called the requirements definition—is a straightforward text report that simply lists the functional and nonfunctional requirements in an outline format. The most obvious purpose of the requirements definition is to provide the information needed by the other deliverables in analysis, which include functional, structural, and behavioral models, and to support activities in the design phase. The most important purpose of the requirements definition, however, is to define the scope of the system. The document describes to the analysts exactly what the system needs to end up doing. When discrepancies arise, the document serves as the place to go for clarification.

**19. What are some of the advantages of using story cards and task lists as a requirements gathering and documentation technique?**

A set of advantages of using story cards and task lists to document requirements is that they are very low tech, high touch, easily updatable, and very portable.

**20. What information is typically included in a system proposal?**

A system proposal brings together into a single comprehensive document the material created during planning and analysis. The system proposal typically includes an executive summary, the system request, the workplan, the feasibility analysis, the requirements definition, and the evolving models that describe the new system. The evolving models include functional models, structural models, and behavioral models. The executive summary provides all critical information in a very concise form. It can be thought of as a summary of the complete proposal.

**21. What is the purpose of the executive summary of the system proposal?**

The executive summary provides all critical information in a very concise form. It can be thought of as a summary of the complete proposal. Its purpose is to allow a busy executive to quickly read through it and determine which parts of the proposal that they need to go through more thoroughly.

# Chapter 4 Business Process and Functional Modeling End-of-Chapter Questions

1. **Why is business process modeling important?**

   Business process models describe the different activities that when combined together support a business process. Business processes typically cut across functional departments (e.g., the creation of a new product will involve many different activities that will combine the efforts of many employees in many departments). Furthermore, from an object-oriented perspective, they cut across multiple objects.

   Business processes is a very constructive activity that can be used to make sense out of the gathered requirements. They are very powerful tools for communicating the analyst's current understanding of the requirements with the user.

2. **How do you create use cases?**

   a) Write each step in the SVDPI form
   b) Make clear the initiator and receiver of action in each step
   c) Write the step from the perspective of an independent observer
   d) Write all steps at the same level of abstraction
   e) Ensure that each use case represents a sensible set of actions
   f) Keep all steps as simple as possible
   g) Repeat all actions as necessary.

3. **Why do we strive to have about three to nine major use cases in a business process?**

   This is about the number of discrete items most people can comfortably deal with at one time. Having fewer major use cases may mean that we are collecting too many activities together into each use case. Having more major use cases may mean that we are describing more than one business process and that we need to distinguish more carefully among business processes before dividing up into sets of use cases.

4. **How do you create use case diagrams?**

   a) Start by drawing the system boundary
   b) Draw the use cases on the diagram
   c) Place the actors on the diagram
   d) Draw the lines connecting the actors to the use cases with which they interact.

5. **How is use case diagramming related to functional modeling?**

   The purpose of use-case models is to provide a description of the problem domain. The use case descriptions and diagrams provide a formal definition of the external behavior or functionality of the business processes. These provide a logical view of the system. In the design phase a physical view of the internals of the new system are modeled.

6. **Explain the following terms: actor, use case, system boundary, and relationship. Use layperson's language, as though you were describing them to a user.**

   a) <u>Actor</u>: The actor is someone (or occasionally some thing) outside the system that provides information or things (inputs) that the system needs, receives some or all of the things or information (outputs) that the system creates, or the actor can both supply and receive things or information from the system.

   b) <u>Use case</u>: A use case is a description of what the system does from the perspective of the user. Since systems can often do many things, the use case often includes a central activity of the system but also shows the many variations that can occur under different circumstances or when varied users interact with it in different ways.

   c) <u>System boundary</u>: The system boundary is the dividing point between those items that are included in the system and those that are not. In terms of information technology, the boundary separates the parts of a job that are done by the application from those that are done by humans. In terms of the entire information system that includes both the humans and applications working together to solve business problems, the boundary can be fuzzier. For the efficient development of new systems, however, the boundary shows what the programmers must include and what they should not include in the framework of the project.

   d) <u>Relationship</u>: Use cases are not independent and stand alone items in most examples. Pairs (or more) of use cases can be related can be associated -- for example there can be communication between an actor and a use case. Use cases can be extended such that a normal flow of events can be extended to another use case showing an alternate or exceptional flow. An include relationship shows that one or more use case is a part of another more encompassing use case. Finally, a generalization relationship shows that a use case is a specialized example of a more general case

**7. Every association must be connected to at least one _______ and one _________. Why?**

Every association must be connected to at least one use **case** and one **actor**. This is the definition of an association—it shows two-way communication between the use case and the actor.

**8. What are some heuristics for creating a use case diagram?**

No order is implied by the diagram, so it is not important to sequence the various use cases. It is usually necessary to redraw the diagram several times to make the diagram easy to read. There should be no more than three to nine use cases on the model to keep it simple yet informative. Place the use-cases to minimize crossing lines.

**9. Why is iteration important in creating use cases?**

Not even the most experienced and talented analysts will be able to describe use cases or diagram them perfectly the first time. Those who turn in a first draft usually have a product that is missing something. If you go into a project expecting to build it up over time and perform many iterations, you will find that the development process goes more smoothly and is less frustrating

**10. What is the purpose of an activity diagram?**

Activity diagrams are used to model the behavior in a business process independent of objects. Activity diagrams can be viewed as sophisticated data flow diagrams that are used in conjunction with structured analysis. Activity diagrams include notation that addresses the modeling of parallel, concurrent activities and complex decision processes. Activity diagrams can be used to model everything from a high-level business workflow that involve many different use cases, to the details of an individual use case, all the way down to the specific details of an individual method. In a nutshell, activity diagrams can be used to model any type of process.

**11. What is the difference between an activity and an action?**

An Activity is used to represent a set of actions. In other words, an activity can be decomposed further into a set of activities and/or actions, whereas an action represents a simple non-decomposable piece of the overall behavior being modeled. An Action is a simple, non-decomposable behavior and is labeled by its name.

**12. What is the purpose of a fork node?**

The fork node is used to split the behavior of the business process into multiple parallel or concurrent flows. Unlike the decision node, the paths are not mutually exclusive, in other words both the paths are executed concurrently.

**13. What are the different types of control nodes?**

There are seven different types of *control nodes* in an activity diagram:

a) Initial: This node portrays the beginning of a set of actions or activities.
b) Final-activity: This node is used to "stop the process" being modeled.
c) Final-flow: Similar to the final-activity node, except that it stops a specific path of execution through the business process, but allows the other concurrent or parallel paths to continue.
d) Decision: This node is used to represent the actual test condition that is used to determine which of the paths exiting the decision node is to be traversed.
e) Merge: This node is used to bring back together multiple mutually exclusive paths that have been split based on an earlier decision
f) Fork: A fork node is used to split the behavior of the business process into multiple parallel or concurrent flows.
g) Join: Finally the join node brings back together the separate parallel or concurrent flows in the business process into a single flow.

**14. What is the difference between a control flow and an object flow?**

A control flow shows:
a) The sequence of execution through a business process
b) It is represented by a solid line with an arrowhead on it showing the direction of the flow
c) These can be attached only to actions or activities.

An object flow shows:
a) The flow of an object from one activity (or action) to another activity (or action)
b) They are depicted as a dashed line with an arrow on it showing the direction of the flow
c) An individual object flow must be attached to an action or activity on one end and an object node on the other end.

**15. What is an object node?**

Object nodes represent the flow of information from one activity to another activity. Activities and actions typically modify or transform objects. Object nodes model these objects in an activity diagram.

- Object node is used to represent an object that is connected to a set of Object Flows
- Object node is labeled by its class name

**16. Explain how a detail use case differ from an overview use case? When are each used?**

The overview use case provides a high level version of the system. It shows the basic information and is created early in the information requirements determination process. In contrast, the detail level use case shows all of the aspects of the use case needed to document the system.

**17. How does an essential use case differ from a real use case?**

The essential use case uses only the minimal essential issues necessary to understand specific functionality. In contrast, the real use case adds in the specifics of how the system implements the request. This seems analogous to the logical versus the physical views of the system.

**18. What are the major elements of an overview use case?**

The major elements are its name, ID number, primary actor, type, and a brief description

**19. What are the major elements of a detail use case?**

The major elements consist of all of the overview use case elements plus a listing of stakeholders and interests; trigger and trigger type; relationships; normal flow of events; listing of sub-flows and alternate/exceptional flows.

**20. What is the viewpoint of a use case, and why is it important?**

The viewpoint is from outside the system. This is important as a way to translate business needs into technical specifications. Ultimately, the business needs are outside the system residing with stakeholders such as customers and employees. It is important to insure that these stakeholders' needs are addressed by the system. Creating the use cases from their perspective is a way to emphasize this business-technology complementarily.

**21. What are some guidelines for designing a set of use cases? Give two examples of the extend associations on a use case diagram. Give two examples for the include association.**

(Assuming that the first question is a broad overview of the chapter) There are several levels of detail from which the student can approach this question. At its highest level, the analyst must consider the major case and its multitude of possible variations; must consider the boundaries, must consider all the stakeholders and all inputs and outputs of the system; and must translate the descriptions accurately into diagrams.

The extend association can include; out-of-stock situation activities during an order; the customer getting a discount for the purchase of two rather than just one of an item in stock.

The uses association can include: a customer uses a shopping cart to gather products from a website shopping center; a "calculate invoice total" use case could use a "look up tax from table" use case.

**22. Which of the following could be an actor found on a use case diagram? Why?**

**Ms. Mary Smith**
**Supplier**
**Customer**
**Internet customer**
**Mr. John Seals**
**Data entry clerk**
**Database administrator**

In the above cases either of the ones listed below could perform the role of an actor.

a) Ms. Mary Smith
b) Customer
c) Internet Customer
d) Mr. John Seals.

The reason being, an *actor* is not a specific user, but a role that a user can play while interacting with the system. An actor can also represent another system in which the current system interacts. In this case, the actor optionally can be represented by a rectangle with <<actor>> and the name of the system. Basically, actors represent the principal elements in the environment in which the system operates. Actors can provide input to the system; receive output from the system, or both. Sometimes an actor plays a specialized role of a more general type of actor. For example, there may be times when a new patient interacts with the system in a way that is somewhat different than a general patient. In this case, a *specialized actor* (i.e., new patient) can be placed on the model, shown using a line with a hollow triangle at the end of the more general actor (i.e., patient). The specialized actor will inherit the behavior of the more general actor and extend it in some way

**23. What is a walkthrough? How does it relate to verification and validation?**

A walkthrough is essentially a peer review of a product. In the case of the functional models, a walkthrough is a review of the different models and diagrams created during functional modeling. This review typically is completed by a team of individuals that comes from the development team and the client. The purpose of a walkthrough is to thoroughly test the fidelity of the functional models to the functional requirements and to ensure that the models are consistent. That is, a walkthrough uncovers errors or faults in the evolving specification. However, a walkthrough does not correct errors—it simply identifies them. Error correction is to be accomplished by the team after the walkthrough is completed.

**24. What are the different roles played during a walk-through? What are their purposes?**

There are specified roles that different members of the walkthrough team can play. The first is the presenter role. This should be played by the individual who is primarily responsible for the specific representation being reviewed. This individual presents the representation to the walkthrough team.

The second role is recorder, or scribe. The recorder should be a member of the analysis team. This individual carefully takes the minutes of the meeting by recording all significant events that occur during the walkthrough. In particular, all errors that are uncovered must be documented so that the analysis team can address them. The third role is to have someone who raises issues regarding maintenance of the representation. Due to the emphasis on reusability in object-oriented development, this role becomes particularly crucial.

Finally, someone must be responsible for calling, setting up, and running the walkthrough meetings.

**25. How are the different functional models related and how does this impact the verification and validation of the models?**

There are three different representations for the functional model: activity diagrams, use-case descriptions, and use-case diagrams. A set of rules have been developed to ensure that these three representations are consistent among themselves.

First, when comparing an activity diagram to a use-case description, there should be at least one event recorded in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description for each activity or action that is included on an activity diagram, and each event should be associated with an activity or action.

Second, all objects portrayed as an object node in an activity diagram must be mentioned in an event in the normal flow of events, subflows, or alternate/exceptional flows of the use-case description.

Third, sequential order of the events in a use-case description should occur in the same sequential order of the activities contained in an activity diagram.

Fourth, when comparing a use-case description to a use-case diagram, there must be one and only one use-case description for each use case, and vice versa.

Fifth, all actors listed in a use case description must be portrayed on the use-case diagram. Furthermore, each one must have an association link that connects it to the use case and must be listed with the association relationships in the use-case description. In some organizations, we should also include the stakeholders listed in the use-case description as actors in the use-case diagram.

Sixth, all other relationships listed in a use-case description (include, extend, and generalization) must be portrayed on a use-case diagram.

Finally, there are many diagram-specific requirements that must be enforced. For example, in an activity diagram a decision node can be connected to activity or action nodes only with a control flow, and for every decision node there should be a matching merge node. Every type of node and flow has different restrictions.

# Chapter 5 Structural Modeling
# End-of-Chapter Questions

1. **Describe to a businessperson the multiplicity of a relationship between two classes.**

   When a relationship exists between classes, the multiplicity documents how many objects may or must be associated between these classes in the form of minimum and maximum numbers. In general, multiplicity reflects business rules regarding the nature of these relationships among classes.

2. **Why are assumptions important to a structural model?**

   The process of developing a structural model involves much learning about the increasing levels of details about a new system. In many cases, the analyst must continue to progress in developing the model, even without knowing all of the business rules. Sometimes the firm may not have encountered situations that cannot be handled informally. Assumptions are important for continuing to work when faced with uncertainty. However, the developer risks having to do a great deal of rework if the assumptions prove to be wrong. Therefore, the developer should check these assumptions at the earliest possible occasion with users and/or sponsors of the project.

3. **What is an association class?**

   These are classes formed when a relationship itself has associated properties, especially when its classes share a many-to-many relationship. In these cases, a class is formed, called an association class that has its own attributes and operations.

4. **Contrast the following set of terms: object, class, method, attribute, super class, subclass, concrete class, abstract class**

   A **class** is a general template that we use to create specific instances, or **objects**, in the application domain. There are two different general kinds of classes of interest during the analysis phase: **concrete** and **abstract**.

   An **attribute** represents a piece of information relevant to the instance/class in the domain of the particular application. **Method** is generally used as another name for operation. Therefore, a method is something that the object can do, a task that can be performed. The term property is not presented as far as I can tell in this chapter, but should be something of a synonym for attribute.

   The **superclass** will contain the attributes and operations shared by a number of subclasses. The **subclasses** inherit the characteristics of the super class and may contain further attributes and operations that belong to each of themselves alone.

   A **concrete class** is a template actually used to create objects. An **abstract class** may be a generalization of multiple concrete classes that is useful for indicating common features/attributes/methods for the individual concrete classes, but is not used to generate objects itself.

5. **Give three examples of derived attributes that may exist on a class diagram. How would they be denoted on the class diagram?**

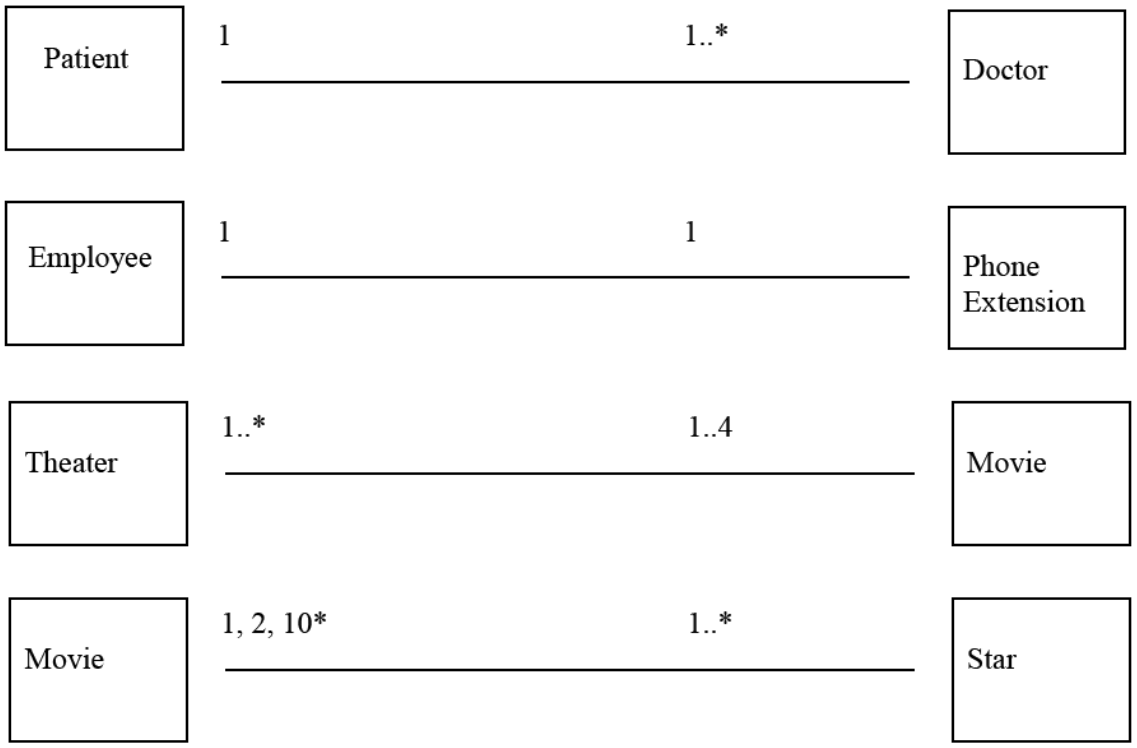
   Derived attributes, which are attributes that can be calculated or derived; these special attributes are denoted by placing a slash (/) before the attribute's name. For example:

   a) -/ age (derived from the difference between birth date and current date.
   b) -/ line-item cost (derived from multiplying product price by quantity)
   c) -/ grade point average (derived from the dividend of total grade points by total units)

6. **What are the different types of visibility? How would they be denoted on a class diagram?**

   Visibility relates to the level of information hiding to be enforced for the attribute. The visibility of an attribute can be public (+), protected (#), or private (-).

7. **Draw the relationships that are described by the following business rules. Include the multiplicities for each relationship.**
   **A patient must be assigned to only one doctor, and a doctor can have one or more patients.**
   **An employee has one phone extension, and a unique phone extension is assigned to an employee.**
   **A movie theater shows at least one movie, and a movie can be shown at up to four other movie theaters around town.**
   **A movie either has one star, two co-stars, or more than 10 people starring together. A star must be in at least one movie.**



8. **How do you designate the reading direction of a relationship on a class diagram?**

   A primary purpose of the class diagram is to show the relationships, or associations, that classes have with one another. When multiple classes share a relationship (or a class shares a relationship with itself), a line is drawn and labeled with either the name of the relationship or the roles that the classes play in the relationship.

   For example, the two classes Patient and Appointment can be associated with one another whenever a patient schedules an appointment. Thus, a line labeled "schedules" connects patient and appointment, representing exactly how the two classes are related to each other. Also there would be a small solid triangle beside the name of the relationship. The triangle allows a direction to be associated with the name of the relationship. (Please refer to Fig. 6-2 in the Chapter)

   One can also infer that while the classes or objects are generally presented from left to right and top to bottom, there really is no definite sequence of these, so one could start anywhere and move in any direction.

9. **For what purpose is an association class used in a class diagram? Give an example of an association class that may be found in a class diagram that captures students and the courses they have taken.**

   The association class is used to show a relationship where the relationship itself has attributes -- in other words which has information regarding the relationship itself that should be captured. Students enroll in courses. There is information about that enrollment itself -- when it occurs and the grade received, for example, that would be of interest to keep track of.

**10. Give two examples of aggregation, generalization, and association relationships. How is each type of association depicted on a class diagram?**

Aggregation: The spark plug is part of the engine; the engine is part of the car
Generalization: The graduate student is a student; the rock music CD is a CD
Association: customer orders book; UPS/FedEx/Post Office delivers book

Generalization shows that one class (subclass) inherits from another class (superclass), meaning that the properties and operations of the superclass are also valid for objects of the subclass. The generalization path is shown with a solid line from the subclass to the superclass and a hollow arrow pointing at the superclass. Generalization relationship occurs when you need to use words such as "is a-kind-of" to describe the relationship.

Aggregation is used when classes actually comprise other classes. For example, think about a doctor's office that has decided to create health care teams that include doctors, nurses, and administrative personnel. As patients enter the office, they are assigned to a health care team that cares for their needs during their visits. Typically, you can identify these kinds of associations when you need to use words like "is a part of" or "is made up of" to describe the relationship.

**11. Identify the following operations as constructor, query, or update. Which operations would not need to be shown in the class rectangle?**
**Calculate employee raise (raise percent)**
**Calculate sick days ( )**
**Increment number of employee vacation days ( )**
**Locate employee name ( )**
**Place request for vacation ( )**
**Find employee address ( )**
**Insert employee ( )**
**Change employee address ( )**
**Insert spouse ( )**

| **Operation** | **Classification** |
|---|---|
| Calculate employee raise (raise percent) | Update |
| Calculate sick days () | Update |
| Increment employee vacation days () | Update |
| Locate employee name () | Query |
| Place request for vacation () | Update |
| Find employee address () | Query |
| Insert employee () | Constructor |
| Change employee address () | Update |
| Insert spouse () | Constructor |

Assuming that spouse is a class; if spouse is an attribute of employee, then, in effect, inserting spouse would be changing an attribute value and more appropriately called “update spouse" or "add spouse-name/spouse identifier”

Constructor is generally available to all classes and not shown.

12. **How are the different structural models related and how does this impact verification and validation of the model.**

Three representations could be used for structural modeling: CRC cards, class diagrams, and object diagrams. Because an object diagram is simply an instantiation of some part of a class diagram, we limit our discussion to CRC cards and class diagrams. We need a set of rules that will test the consistency within the structural models.

First, every CRC card should be associated with a class on the class diagram, and vice-versa.

Second, the responsibilities listed on the front of the CRC card must be included as operations in a class on a class diagram, and vice versa.

Third, collaborators on the front of the CRC card imply some type of relationship on the back of the CRC card and some type of association that is connected to the associated class on the class diagram.

Fourth, attributes listed on the back of the CRC card must be included as attributes in a class on a class diagram, and vice versa.

Fifth, the object type of the attributes listed on the back of the CRC card and with the attributes in the attribute list of the class on a class diagram implies an association from the class to the class of the object type.

Sixth, the relationships included on the back of the CRC card must be portrayed using the appropriate notation on the class diagram.

Seventh, an association class should be created only if there is indeed some unique characteristic (attribute, operation, or relationship) about the intersection of the connecting classes.
Finally, some specific representation rules must be enforced.

**13. How are the different structural models related to the different functional models, and how does this affect verification and validation of the models?**

To balance the functional and structural models, we must ensure that the two sets of models are consistent with each other. That is, the activity diagrams, use-case descriptions, and use-case diagrams must agree with the CRC cards and class diagrams that represent the evolving model of the problem domain. In this case, there are four sets of associations between the models.

First, every class on a class diagram and every CRC card must be associated with at least one use case, and vice versa.

Second, every activity or action contained in an activity diagram and every event contained in a use-case description should be related to one or more responsibilities on a CRC card and one or more operations in a class on a class diagram and vice versa.

Third, every object node on an activity diagram must be associated with an instance of a class on a class diagram and a CRC card or an attribute contained in a class and on a CRC card.

Fourth, every attribute and association/aggregation relationships contained on a CRC card (and connected to a class on a class diagram) should be related to the subject or object of an event in a use-case description.

# Chapter 6 Behavioral Modeling
# End-of-Chapter Questions

1. **How is behavioral modeling related to structural modeling?**

   Behavioral models describe the internal dynamic aspects of an information system that supports the business processes in an organization. Behavioral modeling describes that things that happen and changes that occur during use of a system whereas structural modeling describes the essentially permanent elements of the system.

2. **Describe how the use of activity diagrams with swimlanes are helpful in behavioral modeling.**

   Adding swimlanes to an activity diagram not only shows the various activities that need to be completed, the swimlanes allow them to be assigned particular actors and/or objects.

3. **What is the difference between an external and internal actor?**

   Actors make decisions but need not be human. They can be a person, a company, a computer program, or a computer system. If they are inside the system then they are an internal actor and if they are outside the system, for example a client, they are an external actor.

4. **How does a use case relate to an activity diagram?**

   A use case shows what your system should be doing while an activity diagram allows you to specify how your system will accomplish what it should be doing.

5. **How does a use case scenario relate to an activity diagram?**

   A use case scenario is a single path through the use case while an activity diagram allows you to specify how your system will accomplish what it should be doing.

**6. How does a use case relate to a generic sequence diagram?**

A sequence diagram is a dynamic model that shows the explicit sequence of messages that are passed between objects in a defined interaction. The sequence diagram can be a generic sequence diagram that shows all possible scenarios for a use case.

**7. How does a use case scenario relate to an instance sequence diagram?**

The sequence diagram can be a generic sequence diagram that shows all possible scenarios for a use case, but usually each analyst develops a set of instance sequence diagrams, each of which depicts a single scenario within the use case.

**8. Contrast the following sets of terms: state, behavior, class, object, action, and activity.**

Objects have behaviors that are described by operations. The state of an object at a given point in time is the collection of data available. The purpose of behavior is largely to transition the object from one state to a new state.

The class is a template that generates objects. Objects become available to be instantiated with data that pertain to a particular instance of the class/object.

An action is an atomic, non-decomposable process that cannot be interrupted. They are associated with transitions. In contrast, the activity is non-atomic and decomposable process that is in essence comprised of actions. These are associated with states.

**9. Why is iteration important when creating a behavioral model?**

As we develop, expand, and validate one model, we learn more about other aspects of the system, which helps us elaborate on allied models. As we learn about system behaviors, we come to understand more about the structures needed to support them. We need to update older models to reflect the new learning. In the process of that updating, we may learning more, create more questions, and need to update additional models until the entire set is more or less stable.

**10. What are the main building blocks for the sequence diagram? How they are represented on the model?**

The main elements of the sequence diagram are the actor, object, lifeline, focus of control, message, and object destruction. The actor is the person or system that initiates and derives benefit from the system. Objects participate in the sequence of activities by receiving or sending messages. A lifeline denotes the life of an object from creation to when it is no longer needed. A focus of control shows which objects are sending and receiving messages at a particular time. A message shows that information is being conveyed from one object to another. Object destruction shows when the need for an object is finished and the object is removed from existence.

The actor is represented by a stick figure; the object by a rectangle with the object and class names inside; the lifeline by a vertical dotted line; the focus of control by a thin vertical rectangle; the message by an arrow directed line with the name of the message above. The object destruction is shown by an "x" at the end of an object's lifeline.

**11. How do you show that a temporary object is to go out of existence on a sequence diagram?**

Place an "x" at the bottom of its lifeline to show that a temporary object is to go out of existence on a sequence diagram. A dotted line runs vertically below each actor and object to denote the lifeline of the actors/objects over time. Sometimes an object creates a temporary object, and in this case an X is placed at the end of the lifeline at the point the object is destroyed.

For example, think about a shopping cart object for a Web commerce application. The shopping cart is used for temporarily capturing line items for an order, but once the order is confirmed, the shopping cart is no longer needed. In this case, an X would be located at the point at which the shopping cart object is destroyed.

**12. Do lifelines always continue down the entire page of a sequence diagram? Explain.**

No. When the object continues to exist for the entire sequence of events, then it will continue down to the bottom of the diagram. Otherwise, an "x" shows where it ends.

**13. Describe the steps used to create a sequence diagram.**

1. The context of the sequence diagram must be determined. Usually, this is a scenario from a use case.
2. The objects to be used in the sequence being modeled are identified.
3. The lifeline for each object is set.
4. Messages are added to the diagram.
5. The focus of control on each object's lifeline is drawn in.
6. The sequence diagram is validated to make sure that all steps in the process have been accounted for.

**14. When drawing a sequence diagram, what guidelines should you follow?**

1. Strive for left to right ordering of messages
2. If an actor and object represent the same idea, name them the same.
3. Place the initiator of the scenario on the left of diagram.
4. When there are multiple objects of the same type, be sure to name them.
5. Only show return values when not obvious.
6. Justify message names and return values near the arrowhead.

**15. Are states always depicted using rounded rectangles on a behavioral state machine? Explain.**

Most of the time they are. The exceptions are the initial state and the final state. These are shown with a filled in circle for initial state and a "bullseye" circle for the final state.

**16. What is CRUDE analysis, and what is it used for**

CRUDE analysis uses a CRUDE matrix, in which each interaction among objects is labeled with a letter for the type of interaction: C for create, R for read or reference, U for update, D for delete, and E for execute. Each cell in the matrix represents the interaction between instances of the classes.

A CRUDE matrix is most useful as a system-wide representation. Once a CRUDE matrix is completed for the entire system, the matrix can be scanned quickly to ensure that every class can be instantiated. Each type of interaction can be validated for each class.

**17. What kinds of events can lead to state transitions on a behavioral state machine?**

In the parlance of behavioral state machines, every event leads to a state transition. This is because events are defined as anything that changes a value which in turn describes the state of the behavioral state machine. In other words, transitions occur only as the result of an event. Specific types of events are as follows.

1. Data value change: a change in one of the data values that collectively describe the state of the object. For example, a patient transitions from "new" to "existing" after the first visit.
2. Boolean condition: a Boolean test that is applied at a certain point. For instance, after a patient transitions to "existing" we may ask the question "is the patient insured?" which would lead to state changes based on billing policies.
3. Time lapse: after a certain amount of time in a given state, the object may spontaneously change to a different state. For instance, a patient that is in state "inactive" (perhaps because he/she has no future appointments), may be removed from the system after 5 years.

**18. What are the steps in building a behavioral state machine?**

1. Determine the context of the behavioral state machine, usually a class.
2. Identify the various states that an object will have over its lifetime including the boundaries of initial and final states.
3. Determine the sequence through the states that the object can pass over its lifetime.
4. Identify events, actions, and guard conditions associated with the transition between states.
5. Validate the behavioral state machine that it is possible to reach the final state and that it is possible to leave each state, except the final state.

**19. When drawing a behavioral state machine, what guidelines should you follow?**

1. Only create behavioral state machines for "complex" objects.
2. Draw the initial state in the top left corner of the diagram.
3. Draw the final state in the bottom right corner of the diagram.
4. Use simple, but descriptive, names for states.
5. Question "black hole" and "miracle" states.
6. Guard conditions should be mutually exclusive.
7. Transitions should be associated with messages and operations.

**20. How are guard conditions shown on a behavioral state machine?**

A guard condition is a Boolean expression that includes attribute values, which allows a transition to occur only if the condition is true.

The Boolean expression is placed in brackets and located along the relevant transition.

**21. Describe the type of class that is best represented by a behavioral state machine. Give two examples of classes that would be good candidates for behavioral state machine.**

A class that responds to multiple events and that has significant changes in state over time.

Examples include: invoices and purchase orders; inventories

**22. Identify the model(s) that contain each of the following components: actor, association, class, extends, association, final state, guard condition, initial state, links, message, multiplicity, object, state, transition, and update operation.**

Behavioral State Machine
State, final state, initial state, event, transition, guard condition

Collaboration Diagram
Actor, object, association, message

Sequence Diagram
Actor, object, lifeline, focus of control, message, object destruction

Class Diagram
Class, attributes, operations, associations, multiplicity

CRC Card
Class name, ID, type, description, associations, responsibilities, collaborators, attributes, relationships, aggregation, generalization, extends associations

Use Case
Use case name, ID, importance, actor, stakeholders, trigger and relationships

# Chapter 7 Moving on to Design End-of-Chapter Questions

1. **Explain the primary difference between an analysis model and a design model?**

   The analysis model essentially ignores non-functional requirements such as performance and system environment issues. In contrast, the design models take these issues into account toward preparing for development that is affordable and maintainable as well as meeting functional requirements.

2. **What is meant by balancing the models?**

   Balancing the models is the process of ensuring the consistency among all of the analysis models. For example, do the functional and structural models agree? What about the functional and behavioral models? And finally, are the structural and behavioral models trustworthy?

3. **What are the interrelationships between the functional, structural, and the behavioral models that need to be tested?**

   The activity diagrams, use-case descriptions, and use-case diagrams must agree with the CRC cards and class diagrams that represent the evolving model of the problem domain. Next, the activity diagrams, use-case descriptions, and use-case diagrams must agree with the sequence diagrams, communication diagrams, behavioral state machines, and CRUD matrix. Finally, we need to ensure that the CRC cards and class diagrams agree with the sequence diagrams, communication diagrams, behavioral state machines and the CRUD matrix.

4. **What does factoring mean? How is it related to abstraction and refinement?**

   Factoring is the process of organizing modules (classes or methods) taking into account the possibilities of creating new generalized groupings from a set of classes or methods and differentiating concrete new classes or methods. Abstraction and refinement are two specific methods for factoring -- abstracting common features into a "superclass" and refining classes by creating spin-off concrete "subclasses".

5. **What is a partition? How does a partition relate to collaboration?**

   A partition is like a "subsystem". It represents a collection of modules that are interrelated and somewhat self-contained.

**6. What is a layer? Name the different layers.**

Layers are different elements of the system architecture. The main layers suggested by the book are: foundation, system architecture, human-computer interaction, data management, and problem domain.

**7. What is the purpose of the different layers?**

To successfully evolve the analysis model of the system into a design model of the system, we must add the system environment information. One useful way to do this, without overloading the developer, is to use layers.

**8. Describe the different types of classes that can appear on each of the layers.**

| **Layers** | **Sample Classes** |
|---|---|
| Foundation | Date, Enumeration (They include classes that represent fundamental data types, classes that represent fundamental data structures, sometimes referred to as container classes, and classes that represent useful abstractions, sometimes referred to as utility classes.) |
| Problem Domain | Employee, Customer (Domain classes, further detail the classes so that it will be possible to implement them in an effective and efficient manner. |
| Data Access and Management | DataInputStream, FileInputStream (The types of classes that appear in this layer deal with how objects can be stored and retrieved.) |
| Human–Computer Interaction | Button, Panel (Typical classes found on this layer include classes that can be used to represent buttons, windows, text fields, scroll bars, check boxes, drop-down lists, and many other classes that represent user interface elements.) |
| Physical Architecture | ServerSocket, URLConnection (classes that deal with communication between the software and the computer's operating system and the network) |

**9. What issues or questions arise on each of the different layers?**

The foundation layer is, in many ways, a very uninteresting layer. It contains classes that are necessary for any object-oriented application to exist.
At problem domain layer of the development of our system, we will need to further detail the classes so that it will be possible to implement them in an effective and efficient manner.

The data management layer addresses the issues involving the persistence of the objects contained in the system.

The human–computer interaction layer contains classes associated with the View and Controller idea from Smalltalk. The primary purpose of this layer is to keep the specific user interface implementation separate from the problem domain classes. This increases the portability of the evolving system.

The physical architecture layer addresses how the software will execute on specific computers and networks.

**10. What is a package? How are packages related to partitions and layers?**

A package is a general construct that groups together various UML elements. For example, the package can group use-case diagrams, class and collaboration diagrams, and even other sets of packages.

**11. What is a dependency relationship? How do you identify them?**

A dependency relationship shows that a modification dependency exists between two packages. That is, a change in one package can potentially cause a change in another package. This is very important to know for maintenance, testing, and debugging purposes. A dependency relationship is portrayed by a dashed arrow.

**12. What are the five steps for identifying packages and creating package diagrams?**

1. Set the context for the package diagram.
2. Cluster the classes together into partitions based on the relationships that the classes share.
3. Place the clustered classes together in a partition and model the partitions as packages.
4. Identify the dependency relationshi0ps among the packages.
5. Place the dependency relationships on the evolved package diagram.

**13. What needs to be verified and validated in package diagrams?**

The dependency relationships in a package diagram need to be verified and validated.

**14. When drawing package diagrams, what guidelines should you follow?**

1. Create package diagrams to logically organize designs.
2. Organize package diagrams based on semantic relationships. Use vertical positioning to support inheritance; use horizontal positioning to support aggregation and association.
3. Dependency relationships between packages should reflect the existence of semantic relationships between elements of one package with elements of another package.
4. In the case of use case based package diagrams, be sure to include the actors to portray use case usage.
5. Give packages simple, but descriptive names.
6. Make packages cohesive.

**15. What situations are most appropriate for a custom development design strategy?**

The Boolean expression is placed in brackets and located along the relevant transition.

**16. Describe the difference between inheritance and interaction coupling.**

Coupling refers to how interdependent or interrelated the modules (classes, objects, and methods) are in a system. Interaction coupling deals with the coupling among methods and objects through message passing. Inheritance coupling deals with how tightly coupled the classes are in an inheritance hierarchy.

**17. Describe method cohesion.**

Method cohesion addresses the cohesion within an individual method (i.e., how single-minded a method is). Methods should do one and only one thing. A method that actually performs multiple functions is more difficult to understand.

**18. What is meant by class cohesion? What are the characteristics of ideal class cohesion?**

Class cohesion is the level of cohesion among the attributes and methods of a class (i.e., how single-minded a class is). A class should represent only one thing, such as an employee, a department, or an order. Cohesive class should have these attributes:

1. It should contain multiple methods that are visible outside the class.
2. Each visible method performs only a single function.
3. All methods reference only attributes or other methods defined within the class or one of its superclasses.

**19. Define connascence. How is it related to the ideas of encapsulation, coupling, and cohesion?**

Connascence generalizes the ideas of cohesion and coupling, and it combines them with the arguments for encapsulation. From an object-oriented design perspective, it really means that two modules (classes or methods) are so intertwined that if you make a change in one, it is likely that a change in the other will be required. This is very similar to coupling and, as such, should be minimized. However, when you combine it with the encapsulation levels, it is not quite that simple. You should maximize the cohesion (connascence) within an encapsulation boundary and minimize the coupling (connascence) between the encapsulation boundaries.

**20. What situations are most appropriate for a custom development design strategy?**

When the business need is unique, the firm has the necessary in-house functional and technical experience, they have a skilled project manager and a proven methodology, and the time frame is flexible.

**21. What are some problems with using a packaged software approach to building a new system? How can these problems be addressed?**

Companies must accept the functionality that the system provides and this is rarely identical to what the company would prefer. If the package is large in scope, it could mean substantial change in how the company does business.

These problems can be addressed by using customization to tailor the internals of the package to meet user needs, by workarounds that add custom interfaces to help the system fit into the existing environment, and by object wrappers that are built around legacy or pre-existing systems that will interact with the new package software.

**22. Why do companies invest in ERP systems?**

Over time, some firms fall behind in keeping up with the latest information technologies. They will find themselves with many separate, out-of-date systems that are not well coordinated. The ERP system, if it were to work well, would aid in both updating many old systems as well as enabling better integration among various business units. ERP systems tend to be fairly large and present some significant risks as well as the opportunity for significant benefits to firms that invest in them.

**23. What are the pros and cons of using a workaround?**

On the pro side, the workaround may be a relatively inexpensive solution for making an imperfect but inexpensive software package usable. It allows the firm to retain the use of software already invested in. On the con side, venders may not support the packaged software with the workaround (and the workaround may cause unexpected problems with the software. Moreover, the shift from package to custom software workaround -- particular if there are other systems involved, may degrade performance.

**24. When is outsourcing considered a good design strategy? When is it not appropriate?**

Outsourcing can be an excellent strategy particularly if your firm does not have the technology or expertise to develop a particular system in-house. It may also be less expensive, take advantage of developer learning on other similar systems, and allow more rapid creation of a wider array of IT products for a firm. On the other hand, if the requirements for the project have not been thoroughly examined, there is a good chance that the outsourcer will develop good but inappropriate or incomplete software. There are also risks if the vendor encounters difficulties or perhaps even goes out of business.

**25. What is an object wrapper?**

An object wrapper is essentially an object that "wraps around" a legacy system, enabling an object-oriented system to send messages to the legacy system. Effectively, object wrappers create an application program interface (API) to the legacy system.

**26. What is systems integration? Explain the challenges.**

Systems integration refers to the process of building new systems by combining packaged software, existing legacy systems, and new software written to integrate these. The key challenge in systems integration is finding ways to integrate the data produced by the different packages and legacy systems.

**27. What are the differences between the time and arrangements, fixed-price, and value-added contracts for outsourcing?**

The time and arrangements approach is quite flexible because it charges by the actual time and expenses involved. The fixed price contract sets an agreed-upon price for a particular outcome. The value added contract shares the risk of development between outsourcer and client, but also provides the outsourcer continued benefits once the system is completed.

**28. How are the alternative matrix and feasibility analysis related?**

The alternative matrix is organized to show the value of each alternative relative to each criterion. Generally these criteria include technical, economic and organizational feasibility issues. The alternative matrix is a format that can be used to facilitate the ease of comparing alternatives while performing the feasibility analysis.

**29. What is an RFP? How is this different from an RFI?**

The RFP is a request for proposals. It is a document that solicits proposal from vendors, developers or service providers based on a description of what the intended system is expected to accomplish. The RFP generally also tells vendors or other applicants the basis for selection of the winning proposal.

The RFI is a less intensive but similar effort to interact with potential vendors regarding basic services that can be supplied.

# Chapter 8 Class and Method Design End-of-Chapter Questions

1. **When designing a specific class, what types of additional specification for a class could be necessary?**

    1. Ensure that the classes on the problem domain layer are both necessary and sufficient to solve the underlying problem.
    2. Finalize the visibility of the attributes and methods in each class.
    3. Decide on the signature of each method in every class.
    4. Define any constraints that must be preserved by the objects, for example that values must exist only within a specific range as well as the response should constraints be violated. For example, if employees are only allowed to work up to "60" hours per week and someone attempts to enter "61" for the attribute hours worked, what response should be taken by the system?

2. **What are exceptions?**

    Violations of a constraint are known as exceptions in languages such as Java and C++.

3. **What are constraints? What are the three different types of constraints?**

    Constraints are limitations on the range of values or activities that are acceptable within the business problem domain. The three types are pre-conditions, post-conditions, and invariants.

4. **What are patterns, frameworks, class libraries, and components? How are they used to enhance the evolving design of the system?**

    Patterns are combinations of elements that are found in multiple locations. For example, any business situation updating a master file will have some common procedures for examining the transaction files and changing the status of the master file. These repeating situations allow the reuse of design structures saving time and effort for the project members. Patterns frequently need to be customized for a particular application, so they should be carefully reviewed rather than being mindlessly copied.

    Frameworks are a set of implemented classes that can be used as a basis for implementing an application. These can be used to aid in object persistence and creating inheritance. When the framework changes, the program needs, at a minimum, to be recompiled and probably needs to be updated if there are any changes in dependencies.

Class libraries consist of sets of implemented classes and are designed for reuse. These are similar to frameworks, other than the fact that frameworks tend to be domain specific. Frameworks may, in fact, be built using class libraries.

Components are self-contained encapsulated pieces of software that can be "plugged" into a system to provide a specific set of required functionality.

Each of these approaches is used to streamline the development process by allowing work already completed to be added to the current project.

**5. How are factoring and normalization used in designing an object system?**

Factoring is the process of separating out aspects of a method or class into a new method or class to simplify the overall design. Normalization is a technique borrowed from relational database design aimed at helping to identify new classes and insure that all attributes are dependent on their assigned classes and only on their assigned classes.

**6. What are the different ways to optimize an object system?**

1. Review the access paths between objects.
2. Review the attributes of each class.
3. Review the direct and indirect fan-out of each method.
4. Look at execution order of the statements in often-used methods.
5. Avoid recomputation by creating derived attributes.

**7. What is the typical downside of system optimization?**

The design before optimization is generally aimed at understandability. After optimization, the design is likely to perform better, but the linkage to the problem and user domain is likely to be somewhat more obscure and difficult to explain.

**8. What is the purpose of a contract? How are contracts used?**

A contract formalizes the interactions between client and server. The client object sends a message to the server object which guarantees, given that all constraints are met, the return of the response to the request.

**9. What is the Object Constraint Language? What is its purpose?**

The Object Constraint Language (OCL) is a complete language designed to specify constraints. Essentially, all OCL expressions are simply a declarative statement that evaluates to either being true or false. If the expression evaluates to true, then the constraint has been satisfied.

**10. What is the Structured English? What is its purpose?**

Structured English is simply a formal way of writing instructions that describe the steps of a process (certainly, it can be used to describe constraints). Because it is the first step toward the implementation of the method, it looks much like a simple programming language. Structured English uses short sentences that clearly describe exactly what work is performed on what data.

**11. What is an invariant? How are invariants modeled in a design of a class? Give an example of an invariant for an hourly employee class.**

Invariants model constraints that must always be true for all instances of a class. We attach invariants to the CRC cards or class diagram by adding a set of assertions.

Employee ID (1..1) (unsigned long) {Employee ID = Employee, Get Employee ID()}

**12. Create a contract for a compute pay method associated with an hourly employee class. Specify the preconditions and postconditions using the Object Constraint Language.**

Method name: "compute-pay"

Class name: "hourly employee"

ID: 1

Clients: Hourly employee, perhaps salaried employee

Associated Use Cases: Paying the employees

Description of Responsibilities: Accept the hours, pay rate, and related information. Perform the math needed to return the total pay for the individual employee.

Arguments received: Hours, pay rate, deductions

Type of value returned: Gross wage, Net wage

Pre-conditions: Hours >= 0 && Hours <= 40

Post-conditions: None.

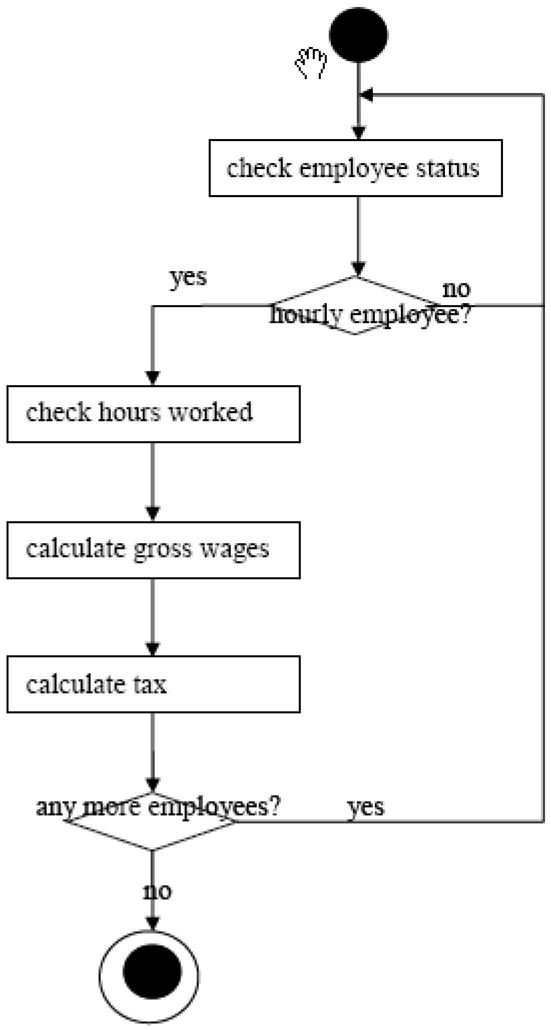
**13. How do you specify a method's algorithm? Give an example of an algorithm specification for a "compute-pay" method associated with an hourly employee class using Structured English.**

Specific algorithms will vary, but should include recognition of some check for hourly employment, some calculation of gross pay, some calculation of net pay, some printing of check. Students may also add in year-to-date information, which is on most pay statements.

```
FOR all Employees in Employee Object Store
    IF Employee Status = "hourly"
    THEN
Perform range check for "hours worked"
Gross wage = "hours worked" * "pay rate"
            Accumulated payroll = accumulated payroll + gross wage
Perform calculate taxes
            Perform sum deductions
            Net wage = gross wage - deductions
            Save totals to Payroll Object Store
            Print pay statement (gross wage, all deductions, net wage)
    ELSE
            Continue to next Employee Object
    ENDIF
```

**14. How do you specify a method's algorithm? Give an example of an algorithm specification for a compute pay method associated with an hourly employee class using an activity diagram.**

Specific algorithms will vary, but should include recognition of some check for hourly employment, some calculation of gross pay, some calculation of net pay, some printing of check. Students may also add in year-to-date information, which is on most pay statements.



15. **How are methods specified? Give an example of a method specification for a compute-pay method associated with an hourly employee class.**

    There is no formal syntax;

    Student should expand the information presented in question 25 adding information such as programmer, date due, programming language.

    Triggers/Events: should indicate a time period such as weekly, bi-weekly, semi-monthly, monthly, or some other reasonable period.

    Algorithm specification should make use of the answer to question 24.

# Chapter 09 Data Management Layer Design
# End-of-Chapter Questions

**1. Describe the steps in object persistence design.**

The four-step approach for object persistence design is:

1. Selecting the format of the storage
2. Mapping the problem domain objects to the object-persistence format
3. Optimizing the object-persistence format
4. Designing the data access and manipulation classes necessary to handle the communication between the system and the database.

**2. How are a file and database different from each other?**

Files are essentially an electronic list of information that is formatted for a particular transaction. Any programs that are written must be developed to work with the file exactly as it is laid out. If there is a need to combine data in a new way, a new file must be created (usually by extracting data from other files) and a program written to work specifically with that new file. Databases, on the other hand, are made up of a collection of data sets that are related to each other in some way. Database management system software creates these data groupings. The DBMS provides access to the data and can usually provide access to any desired subset of data. It is not necessary to write new programs to build a new file in order to retrieve data from the database in a new way.

**3. What is the difference between an end-user database and an enterprise database? Provide an example of each one.**

An end-user database is one that is designed to run on a PC and is used to create personal database applications. An end-user in sales might develop a Microsoft Access database, for example, to keep track of current and prospective client contacts. An enterprise database is one that is capable of handling huge volumes of information for the entire organization. Applications that serve the entire enterprise can be built upon these enterprise databases. These databases are fast, high capacity, but also complex. Oracle is a vendor of enterprise database management systems.

**4. What are the differences between sequential and random access files?**

Sequential access files allow only sequential file operations to be performed because they actually store the data in sequence according to an identifier. These are very efficient when reports are needed for the whole set of data, but are quite inefficient in locating a specific object of interest. In contrast a random access file is optimized for location of specific objects of interest but are less efficient for reports regarding the entire set of data. Generally, these are stored according to some formula or algorithm rather than based on the sequence of some identifier.

**5. Name five types of files and describe the primary purpose of each file.**

1. Master files store the business's or application's core data. The data in a master file is considered fairly permanent, does not change much, and is usually retained for long periods.
2. Look-up files contain reference information that is used primarily during validation processing. A list of valid code values for a field might be referred to during data entry to ensure that a valid code was entered.
3. Transaction files contain information that will be used to update a master file. These files are usually temporary in nature; they are used to collect transactions, the transactions update the master file, and then the transaction files are archived.
4. Audit files are used to help trace changes made to files. An image of a record before and after a change may be written to an audit file so that the change is documented.
5. History files serve as archives for older information that is rarely used. These files can be accessed if necessary, but are usually stored on tape to remove the little-used data from the active data storage areas.

**6. What is the most popular kind of database today? Provide three examples of products that are based on this database technology**

Relational databases are most popular today due to their ease of use and conceptual simplicity. Examples of relational DBMSs on the market include MS Access, Oracle, DB2, Sybase, Informix, and MS SQL Server.

**7. What is referential integrity and how is it implemented in a relational database?**

Referential integrity refers to the need to make sure that the values linking the table together through the primary and foreign keys are valid and correctly synchronized. For example, if a customer is placing an order, we need to have information on the customer in the customer table, The RDBMS will check to see if there is a record for that customer in the Customer table before it will let an order be entered. Checking for known required relationships helps assure referential integrity.

**8. List some of the differences between an ORDBMS and a RDBMS.**

The ORDBMS is an RDBMS with the addition of extensions to handle the storage of objects in the relational table structure. This is done by allowing user defined data types (such as sound, video, etc.) The ORDBMS can share strengths of searching and data handling with the RDBMS, but typically does not implement all object-oriented features such as multi-valued attributes, repeating groups or inheritance.

**9. What are the advantages of using an ORDBMS over a RDBMS?**

These systems are able to handle complex data without losing the advantages of being based on proven technologies.

**10. List some of the differences between an ORDBMS and an OODBMS?**

The OODBMS leaves the relational model aside. It does implement something called an extent which functions much like a table. While the system generates a unique identifier for each object, it is still good to have an attribute with a unique value for each instance. The OODBMS provide some language dependent form of inheritance and some complex data forms such as repeating groups and multi-valued attributes.

**11. What are the advantages of using an ORDBMS over an OODBMS?**

Being based on established technology, the systems tend to be more mature and have a broader range of personnel with appropriate skill sets.

**12. What are the advantages of using an OODBMS over a RDBMS?**

These directly implement the object-oriented approach and are helpful in participating in a fully object-oriented processing environment. They also can more effectively handle complex data such as in multimedia applications

**13. What are the advantages of using an OODBMS over an ORDBMS?**

The OODBMS directly supports object-oriented approach and can be more tuned to these sorts of applications.

**14. What are the factors in determining the type of object persistence format that should be adopted for a system? Why are these factors so important?**

The data types supported, the types of application systems supported, the existing storage formats, the future needs, the amount of risk the organization is willing to absorb, and the availability of trained personnel. Each of these factors influences the cost, time required, and complexity of proposed systems. There are both short and long range costs to consider as well as the reliability and effectiveness of the information systems based on these various approaches.

**15. Why should you consider the storage formats that already exist in an organization when deciding upon a storage format for a new system?**

This factor is important because the project team needs to be aware of the existing base of technical skills that are available to work with the data storage format. If a data storage format is chosen that is new to the organization, then the team must allocate training and learning time into the project schedule.

**16. When implementing the object persistence in an ORDBMS, what types of issues must you address?**

The mapping between problem domain layer and the data management layer is not straightforward with the ORDBMS. First, the mapping will vary with the specific language and tools used. There is a need to map the problem domain classes to both data management classes and tables in the data management layer. There are nine rules that should be followed for implementing this mapping that mostly focus on dealing with complex data forms such as repeating groups of attributes.

**17. When implementing the object persistence in an RDBMS, what types of issues must you address?**

The mapping between problem domain layer and the data management layer is also more complex with the RDBMS. Essentially, the problem domain layer objects are mapped to relational tables. The general rules for optimizing RDBMS in the non object-oriented environment apply -- such as removing redundant data and minimizing null values. Normalization is also important.

**18. What are some of the nonfunctional requirements that can influence the design of the data management layer?**

- Operational Requirements: DAM layer technologies that must be used
- Performance Requirements: DAM layer speed and capacity
- Security Requirements: Access controls, encryption, and backup
- Political & Cultural Requirements: Date formats, currency conversions

**19. What is the primary purpose of the data access and manipulation classes?**

The primary purpose is to design a set of data access and manipulation classes to ensure the independence of the problem domain classes from the storage format. The data access and manipulation classes handle all communication with the database, in other words they act as a translator between the object persistence and the problem domain objects. In this manner, the problem domain is decoupled from the object storage allowing the object storage to be changed without impacting the problem domain classes.

**20. Why should data access and manipulation classes be dependent on the associated problem domain classes instead of the other way around?**

It is recommended that the data management functionality specifics, such as retrieval and updating of data from the object storage, be included only in classes contained in the Data Management layer. This will ensure that the Data Management classes are dependent on the Problem Domain classes and not the other way around. Furthermore, this allows the design of Problem Domain classes to be independent of any specific object persistence environment, thus increasing their portability and their potential for reuse. This one also implies additional processing. However, the increased portability and potential for reuse realized should more than compensate for the additional processing required.

**21. Why should the object persistent classes be dependent on the associated problem domain classes instead of the other way around?**

It is recommended that the data management functionality specifics, such as retrieval and updating of data from the object storage, be included only in classes contained in the Data Management layer. This will ensure that the Data Management classes are dependent on the Problem Domain classes and not the other way around. Furthermore, this allows the design of Problem Domain classes to be independent of any specific object persistence environment, thus increasing their portability and their potential for reuse. This one also implies additional processing. However, the increased portability and potential for reuse realized should more than compensate for the additional processing required.

# Chapter 10 Human Computer Interaction Layer Design End-of-Chapter Questions

**1. Explain three important user interface design principles.**

The authors list six principles of user interface design:

1. Layout - the interface should be a series of areas on the screen that are used consistently for different purposes.
2. Content Awareness - the user is always aware of where they are in the system and what information is being displayed.
3. Aesthetics - interfaces should look inviting and should be easy to use.
4. User Experience - experiences users prefer ease of use, while inexperienced users prefer ease of learning.
5. Consistency - users can predict what will happen before a function is performed.
6. Minimize Effort - interface should be simple to use.

**2. What are three fundamental parts of most user interfaces?**

1. Navigation mechanism - the way the user gives instructions to the system and tells it what to do.
2. Input mechanism - the way in which the system captures information.
3. Output mechanism - the way the system provides information to the user or to other systems.

**3. Why is content awareness important?**

Content awareness means that the interface makes the user aware of the information delivered through the interface with the least amount of user effort. This is important because if the user is constantly aware of where he is and what he is seeing, he will find the system much easier to use and his satisfaction will be high.

**4. What is white space and why is it important?**

White space refers to areas on an interface that are intentionally left blank. The more white space on an interface, the less dense the information content. Designers need to try and strike a balance between information content and white space. Some white space is necessary to help the users find things on the interface. Generally, more experienced users need less white space than novice users.

**5. Under what circumstances should densities be low? High?**

Low densities are preferred by infrequent or novice users of an interface. These users will be unfamiliar with the interface and will be helped by having a balance of information and white space on the interface. High densities can be acceptable to experienced users of the interface, because they are highly familiar with the information on the interface and do not need as much white space to help them find what they are looking for.

**6. How can a system be designed to be used by experienced and first time users?**

Experienced users prefer systems that focus on ease of use, while novice users prefer systems that are easy to learn. These two goals are not necessarily mutually exclusive. Generally, systems should be set up so that the commonly used functions can be accessed quickly, pleasing the experienced users. To assist the novice users, guidance should be readily available, perhaps through the "show me" functions that demonstrate menus and buttons.

**7. Why is consistency in design important? Why can too much consistency cause problems?**

Consistency means that all parts of the same system work in the same way. This enables the users to predict what will happen because a function in one part of the system works the same way in other parts of the system. Users will be confident as they work with different parts of the system if they can predict the behavior of functions throughout the system. The problem with too much consistency is that sometimes the users don't differentiate forms or reports that look very similar to each other, and inadvertently use the wrong one. So, in these cases, there should be enough unique characteristics to distinguish each form and report from the others.

**8. How can different parts of the interface be consistent?**

The navigation controls can be consistent, using the same icon or command to trigger an action throughout the system. Terminology can be consistent throughout the interface. The content portion of the screen that contains forms and reports should also present consistently designed reports and forms. Messages and information in the status area should be specified consistently throughout the system.

**9. Describe the basic process of user interface design.**

First, identify 'use cases' that describe commonly used patterns of actions that users will perform. These use cases will be valuable in ensuring that the interface permits the users to enact these use cases quickly and smoothly. Next, develop the interface structure diagram, defining the basic structure of the interface (screens, forms, and reports) and how the interface components connect. Third, develop interface standards, the basic design elements that will be used throughout the interface. Fourth, create prototypes of the various interface components (navigation controls, input screens, output screens, forms, and reports). Finally, evaluate the prototypes and make changes as needed.

**10. What are use scenarios and why are they important?**

Use scenarios describe commonly used patterns of actions that users will perform. Use cases describe how users will interact with the system. Use cases are developed for the most common ways of working through the system. These use cases will be valuable in ensuring that the interface permits the users to enact these use cases quickly and smoothly

**11. What is a WND and why is it used?**

A window navigation diagram defines the basic structure of the interface. These diagrams show all the screens, forms, and reports in the system, how they are connected, and how the user moves from one to another. The diagram helps depict the basic components of the interface and how they work together to provide users the needed functionality. The structure of the interface depicted in the WND can be examined using the use cases to see how well the use cases can be performed. This is an important early step in developing simple paths through the most common activities performed in the system.

**12. Why are interface standards important?**

Interface standards help define the basic, common design elements in the system. These standards help ensure consistency throughout the system.

**13. Explain the purpose and contents of interface metaphors, interface objects, and interface actions, interface icons, and interface templates.**

- The interface metaphor provides a concept from the real world that helps the user understand the system and how it works. If the user understands the metaphor being used, he will probably be able to predict where to find things and how things will work even without actually using the system.
- Interface objects are the fundamental building blocks of the system. Object names should be based on the most understandable terms.
- Interface actions specify the navigation and command language style and the grammar of the system. Action terminology is also defined.
- Interface icons are pictures that are used to represent objects and actions in the system, often shortcuts, that are available throughout the system.
- The interface template defines the general appearance of all screens in the information system and all forms and reports that are used. The template consolidates all the other major interface design elements - metaphors, objects, actions, and icons.

14. **Why do we prototype the user interface design?**

Prototyping helps the users and programmers understand how the system will perform. Prototypes can be very useful in helping the users conceptualize how they will actually work with the system, and prototypes can help identify problems or misconceptions in the interface before it is actually implemented.

**15. What are Krug's three design principles?**

First, the user should never have to think about how to navigate the user interface. Second, he suggests that the number of clicks that a user must perform to complete their task is somewhat irrelevant. Third, minimize the number of words on the screen.

**16. Describe three basic principles of navigation design.**

The navigation component of the interface enables the user to enter commands to navigate through the system and perform actions to enter and review information it contains. The three basic principles of navigation design are:

1. Prevent Mistakes: The first principle of designing navigation controls is to prevent the user from making mistakes. Mistakes can be reduced by labeling commands and actions appropriately and by limiting choices.
2. Simplify Recovery from mistakes: No matter what the system designer does, users will make mistakes. The system should make it as easy as possible to correct these errors.
3. Use consistent grammar order: One of the most fundamental decisions is the grammar order. The grammar order should be consistent throughout the system, both at the data element level as well as at the overall menu level.

**17. How can you prevent mistakes?**

Mistakes can be reduced by labeling commands and actions appropriately and by limiting choices. Too many choices can confuse the user, particularly when they are similar and hard to describe in the short space available on the screen. When there are many similar choices on a menu, consider creating a second level of menu or a series of options for basic commands.

**18. Explain differences between object-action order and action-object order.**

The fundamental goal of the navigation design is to make the system as simple to use as possible, by preventing the user from making mistakes, simplifying the recovery from mistakes, and using a consistent grammar order. One of the most fundamental decisions is the grammar order. Most commands require the user to specify an object (e.g., file, record, word), and the action to be performed on that object (e.g., copy, delete). The interface can require the user to first choose the object and then the action (an object-action order), or first choose the action and then the object (an action-object order).

**19. Describe the four types of navigation controls.**

There are two traditional hardware devices that can be used to control the user interface: the keyboard and a pointing device such as a mouse. There are three basic software approaches for defining user commands: languages, menus, and direct manipulation.

1. Languages: With a command language, the user enters commands using a special language developed for the computer system (e.g., UNIX and SQL both use command languages). Command languages sometimes provide greater flexibility than other approaches because the user can combine language elements in ways not predetermined by developers.
2. Menus: The most common type of navigation system today is the menu. A menu presents the user with a list of choices, each of which can be selected. Menus are easier to learn than languages because a limited number of available commands are presented to the user in an organized fashion.
3. Direct Manipulation: With direct manipulation, the user enters commands by working directly with interface objects.

**20. Why are menus the mostly commonly used navigation control?**

The most common type of navigation system today is the menu. A menu presents the user with a list of choices, each of which can be selected. Menus are easier to learn than languages because a limited number of available commands are presented to the user in an organized fashion. Clicking on an item with a pointing device or pressing a key that matches the menu choice (e.g., a function key) takes very little effort. Therefore, menus are usually preferred to languages.

**21. Compare and contrast four types of menus.**

The types of menus are Menu bar, drop-down menu, pop-up menu, tab menu, tool bar, and image map.

Menu bar: This is the main menu of the system. It gives a list of commands at the top of the screen and is always displayed on the screen.

- Use the same organization as the operating system and other packages.
- Menu items are always one word, never two.
- Menu items lead to other menus rather than perform action.
- Never allow users to select actions they can't perform

Drop-down menu: This is the second level of menu often from the main menu. This is the menu that drops down immediately below another menu and disappears after one use.

- Menu items are often multiple words
- Avoids abbreviations
- Menu items perform action or lead to another cascading drop-down menu, pop-up menu, or tab menu.

Pop-up Menu: This mainly serves as a short cut to commands for experienced users. This menu pops up, floats over the screen and then disappears after one use.

- Pop-up menus often (not always) invoked by a right click in Windows-based systems.
- These are often overlooked by novice users so usually they should duplicate functionality provided in other menus.

Tab Menu: This menu is often used when the user needs to change several settings or perform certain related commands. This is a multipage menu with one tab for each page that pops up and floats over the screen and continues to remain on the screen until it is closed.

- Menu items should be short to fit on the tab label.
- Avoid more than one row of tabs, because clicking on a tab to open it can change the order of the tabs and in virtually no other case does selecting from a menu rearrange the menu itself.

**22. Under what circumstances would you use a drop down menu versus a tab menu?**

The drop down menu can be used if the user is new or inexperienced. This is mainly used since the tab menu is used more often when the user needs to change several settings in order to perform several related commands and more over since it is multi-page menu, this could often lead to confusion when the user is new.

Drop down menu being the second level of menu is more elaborate and has no abbreviations in it and it performs action or leads to another cascading drop down, pop up or tab menu. This menu is comparatively much more user friendly as compared to the tab menu.

**23. Under what circumstance would you use an image map versus a simple list menu?**

An image map is used only when the graphic image adds meaning to the menu. In other words, it is commonly used when a graphic image has certain area linked to the actions or other menus.

The list menu is the second level menu after the menu bar and performs action or leads to another cascading drop-down menu, pop-up menu or tab menu.

**24. Describe five types of messages.**

Messages are the way in which the system responds to a user and informs him or her of the status of the interaction. There are 5 types:

1. <u>Error message</u>: This pops up when the user does something that is not permitted to inform the user that he or she has attempted to do something to which the computer cannot respond.
2. <u>Confirmation message</u>: When a user selects a potentially dangerous operation such as deleting a file, a confirmation message asks the user if he or she is sure of performing this operation.
3. <u>Acknowledgment message</u>: Seldom used this informs the user that the operation or task was successfully completed.
4. <u>Delay message</u>: More common when the action requested is delayed and informs the user that the computer is working properly.
5. <u>Help message</u>: This is present in all systems and provides additional information about the system and its components.

**25. What are the key factors in designing an error message?**

All messages should be crafted with care, especially the error messages. The key factors, which need to be kept in mind while designing an error message, are:

- Should always explain the problems in polite succulent terms
- Should explain corrective action as clearly and as explicitly as possible
- In case of complicated errors, the error message should display what the user has entered and suggest probable causes for error.
- When in doubt provide either more information than the user needs or ability to get additional information.
- Error message should provide error number

**26. What is context-sensitive help? Does your word processor have context-sensitive help?**

Context sensitive help provides information that is dependent on what the user was doing when the help was requested. Word processors do have context sensitive help.

**27. How do an essential use case and a real use case differ?**

An essential use case is one that only describes the minimum essential issues necessary to understand the required functionality. A real use case will go further and describe a specific set of steps. The primary difference is that essential use cases are implementation independent, while real use cases are detailed descriptions of how to use the system once it is implemented. As such, real use cases tend to be used only in detailed design, implementation, and testing.

**28. What is the relationship between essential use cases and use scenarios?**

An essential use case is a major process that the system will perform that benefits an actor in some way and is labeled using a descriptive verb-noun phase. There are times when a use case includes, extends, or generalizes the functionality of another use case on the diagram.

On examining the use case and sequence diagrams analysts interviewed the users to develop user scenarios that describe commonly employed patterns of actions the users will perform so the interface enables users to quickly and smoothly perform these scenarios. A use scenario is an outline of the steps that the users perform to accomplish some part of their work. A use scenario is one path through an essential use case.

Use scenarios are presented in a simple narrative description that is tied to the essential use cases developed during the analysis phase. The key point with using use cases for interface design is not to document all possible use scenarios within a use case. The goal is to document two or three of the most common use scenarios so the interface can be designed to enable the most common uses to be performed simply and easily.

**29. What is the relationship between real use cases and use scenarios?**

A use scenario is an outline of the steps that the users perform to accomplish some part of their work. A use scenario is one path through an essential use case. Real use cases are derived from the essential use cases. The design of the navigation for a system is done through the use of window navigation diagrams (WND) and real use cases. Real use cases are derived from the essential use cases, use scenarios, and WNDs. A real use case describes a specific set of steps that a user performs to use a specific part of a system. As such, real use cases are implementation dependent.

**30. Explain three principles in the design of inputs.**

Input design means designing the screens used to enter the information, as well as any forms on which users write or type information (e.g., timecards, expense claims). The goal of the input mechanism is to simply and easily capture accurate information for the system. The fundamental principles for input design reflect the nature of the inputs (whether batch or online) and ways to simplify their collection.

1. Online versus Batch Processing: There are basically two methods of entering inputs into a system online processing and batch processing. With online processing or transaction processing each input item is entered into the system individually usually at the same time as the event or transaction prompting the input. With batch processing, all the inputs collected over some time period are gathered together and entered into the system at one time in a batch.
2. Capture data at the source: Perhaps the most important principle of input design is to capture the data in an electronic format at its original source or as close to the original source as possible.
3. Minimize Keystrokes: Another important principle is to minimize keystrokes. Keystrokes cost time and money, whether a customer, user, or trained data-entry operator performs them.

**31. Compare and contrast batch processing and online processing. Describe one application that would use batch processing and one that would use online processing.**

There are basically two methods of entering inputs into a system online processing and batch processing. With online processing or transaction processing each input item is entered into the system individually usually at the same time as the event or transaction prompting the input. Online processing is most commonly used when it is important to have real-time information about the business process. For example, when you reserve an airline seat, the seat is no longer available for someone else to use.

With batch processing, all the inputs collected over some time period are gathered together and entered into the system at one time in a batch. Some business processes naturally generate information in batches. For example, most hourly payrolls are done using batch processing because time cards are gathered together in batches and processed at once. Batch processing also is used for transaction processing systems that do not require real-time information.

**32. Why is capturing data at source important?**

Perhaps the most important principle of input design is to capture the data in an electronic format at its original source or as close to the original source as possible. In the early days of computing, computer systems replaced traditional manual systems that operated on paper forms. Many business processes still operate this way today. For example, most organizations have expense claim forms that are completed by hand and submitted to an accounting department, which approves them and enters them into the system in batches. There are three problems with this approach. *First*, it is expensive because it duplicates work (the form is filled out twice, once by hand, once by keyboard). *Second*, it increases processing time because the paper forms must be physically moved through the process. *Third*, it increases the cost and probability of error, because it separates the entry from the processing of information; someone may misread the handwriting on the input form, data could be entered incorrectly, or the original input may contain an error that invalidates the information.

**33. Describe four devices that can be used for source data automation.**

Source data automation refers to using special hardware devices to automatically capture data without requiring anyone to type it. Four devices that can be used are:

1. Bar codes: These automatically scan products and that enter data directly into the computer system.
2. Optical character recognition: This can read printed numbers and text. (e.g., checks)
3. Magnetic stripe readers: This can read information encoded on a stripe of magnetic material similar to a diskette. (e.g., credit cards)
4. Smart cards: These contain microprocessors, microchips and batteries, very similar to credit card sized calculators.

**34. Describe five types of input.**

There are many different types of inputs:

1. Text: As the name suggests, a text box is used to enter text. Text boxes can be defined to have a fixed length or can be scrollable and can accept a virtually unlimited amount of text. In either case, boxes can contain single or multiple lines of textual information.

2. Numbers: A number box is used to enter numbers. Some software can automatically format numbers as they are entered, so that 3452478 becomes $34,524.78. Dates are a special form of numbers that sometimes have their own type of number box.

3. Selection Boxes: A selection box enables the user to select a value from a predefined list. The items in the list should be arranged in some meaningful order, such as alphabetical for long lists, or in order of most frequently used. The default selection value should be chosen with care. There are different types of Selection boxes

   - Check boxes
   - Radio buttons
   - Onscreen list boxes
   - Drop-down list boxes
   - Combo boxes
   - Sliders

**35. Why is input validation important?**

Input validation is important as all data which is entered into the system needs to be validated in order to ensure their accuracy.

**36. Describe five types of input validation methods.**

There are six types of validation methods. Five of which are described below:

1. Completeness check: When several fields need to be entered before the form can be processed, completeness check ensures that all required data have been entered. If the completeness check is not done and the information provided id incomplete then the form is returned to the user unprocessed.
2. Range check: A range check permits only numbers between correct values. It ensures that all numeric data entered are within correct minimum and maximum values.
3. Consistency check: Data fields are often interrelated. When this happens, consistency check ensures that the combination of the data is valid. Although it is impossible for the system to know which data are incorrect, it can report the error to the user for correction.
4. Format check: When fields are numeric and contain coded data, format check ensures that the data are of right type. Ideally numeric fields should not permit users to type text data but if this is not possible, then the data entered should be checked to ensure that it is numeric.
5. Database checks: Usually data are compared against information in the database to ensure that they are correct. When this occurs, the database check ensures that the comparison of the data against the data base is correct.

**37. Describe how invariants, preconditions, and postconditions are useful in input validation.**

Input validation, which is also known as data validation, is the process of approving any input supplied by a user. Invariant enforcement happens before that state has been reached. A precondition is a constraint that must be met for a method to execute. A postcondition is a constraint that must be met after the method executes, or the effect of the method execution must be undone. By reviewing the contracts and assertions that captured the constraints, most, if not all, data validation issues can be identified.

**38. Explain three principles in the design of outputs.**

The fundamental principles for output design reflect how the outputs are used and ways to make it simpler for users to understand them.

Understand Report Usage: The first principle in designing reports is to understand how they are used. Reports can be used for many different purposes. In some cases—but not very often— reports are read cover to cover because all information is needed.

Manage information load: Most managers get too much information, not too little (i.e., the information load that the manager must deal with is too great). The goal of a well-designed report is to provide all of the information needed to support the task for which it was designed. This does not mean that the report needs to provide all the information available on the subject—just what the users decide they need in order to perform their jobs.

Minimize Bias: No analyst sets out to design a biased report. The problem with bias is that it can be very subtle; analysts can introduce it unintentionally. Bias can be introduced by the way in which lists of data are sorted because entries that appear first in a list may receive more attention than those later in the list.

**39. Describe five types of outputs.**

There are many different types of reports, such as detail reports, summary reports, exception reports, turnaround documents and graphs.

1. Detailed report: Lists detailed information about all the items requested.
2. Summary report: Lists summary information about all items
3. Turn around document: Turn around documents is a special type of report that is both outputs and inputs. In other words, outputs which turn around and become inputs.
4. Graphs: Charts that are used in addition to and instead of tables of numbers.
5. Exception report: Lists detailed information about certain specific items.

**40. What do you think are three common mistakes that novice analysts make in navigation design?**

One of the hardest things about using a computer system is learning how to manipulate the navigation controls to make the system do what you want. Analysts usually think that:

1. The users have read the manual
2. The users have attended proper training classes
3. The users can seek or rather have external help readily available.

**41. What do you think are three common mistakes that novice analysts make in input design?**

1. Failure to perform proper input validation.
2. Use of overly cumbersome forms
3. Using the wrong controls for a given input item

**42. What do you think are three common mistakes that novice analysts make in output design?**

1. Presenting the user with too much information
2. Using reports that are inconsistent with existing business processes
3. Using reports that are difficult for the user to read

**43. What are the six challenges you face when developing mobile applications?**

Tidwell identifies six challenges that a mobile user interface designer must face. First, the screen size of a phone is tiny. There simply is not a lot of “real estate” available to use (see Figure 10-B). Second, not only are the screens tiny, but they come in different sizes. What works on one screen, may not work on another screen. Third, some screens have haptic abilities, i.e., they respond to touch, orientation, and in some case, they vibrate. Obviously, these abilities are not available on all mobile devices. However, they do provide interesting possibilities for user interface design. Fourth, virtual and actual physical keypads are tiny. Consequently, too much typing can be challenging for the user to input the right information. Fifth, people use their mobile devices, especially their phones, in all kinds of environments. They use them in dark places (like a poorly lit classroom). They use them in bright sun light. They use them in quiet places (like the library or movie theater) and they use them in noisy places (such as at a Virginia Tech football game). These devices are simply used everywhere today. Sixth, because these devices are used everywhere, the users can be easily distracted from the device.

**44. What are the six suggestions to address the mobile computing challenges?**

Based on these challenges, Tidwell provides a set of suggestions that you should follow in designing a user interface for these devices. First, given the mobile context, you really need to focus on what the user needs and not what the user may want. In other words, you really should go back to business process and functional modeling (Chapter 4). In this case, only focus on the tasks that the user will need to perform when they are in the mobile context. This is a good example of a non-functional requirement (mobile computing) impacting the possible functional requirements.

Second, if you are porting an application or web site to a mobile device, remove all "fluff" from the site. By that we mean, strip the site down to its bare essentials. If for some reason, the user needs access to the full site, be sure to provide a link to it in an obvious location. Alternatively, you could provide a complete mobile version of the application or web site to the user. Obviously, the design of the user interface will be different, but the functionality should be the same.

Third, whenever possible, take advantage of the unique capabilities built into these devices. Some of the devices will have GPS built-in. Depending on your application, knowing where the user is could change the results. In other cases, devices such as the iPadTM, have an accelerometer that allows the app to "know" the orientation of the device. Many of devices have speech recognition capabilities, cameras that can be used for scanning, touch screens that allow sophisticated gestures to be used, and haptic feedback, such as bumps and vibrations. All of these capabilities could prove useful in developing different mobile applications.

Fourth, when considering a phone, you tend to have a limited width from which to work. Consequently, you should try to linearize the content of the application (see Figure 10-25). By that we mean, take advantage of vertical scrolling and try to minimize, if not eliminate, horizontal scrolling. It is simply more natural for users to scroll up and down instead of left to right on these devices.

Fifth, optimize your mobile application for the user. This will include minimizing the number of times the device must interact with a server to download or upload information with a server. Not everyone will have access to 3G, alone true 4G, networks. In many cases, uploading and downloading is still very slow. Optimization also includes the user's interaction with the device. Instead of using a lot of typing, scrolling, and taps on a touch screen, consider using the speech recognition capability. It's a lot easier to speak slowly to a smartphone than it is to have to type a lot into a virtual or physical keyboard.

Finally, Tidwell provides a set of reusable patterns that have been customized for mobile devices. These include things such as a vertical stack, filmstrip, and bottom navigation to name a few.

**45. What are the unique navigation controls, input mechanisms, and outputs that mobile computing supports?**

When it comes to the navigation part of the user interface, the primary additional option is the use of the touchscreen. In fact, there is an entire vocabulary when it comes to the way users interact with touchscreens. Today, the designer needs to consider tapping, pinching, spreading, flicking, scrolling (one-finger vs. two-finger), and dragging to name a few.

For the input part of the user interface, the primary additional options to consider are the use of the camera as a device to scan items, the use of the microphone as a speech input device, and the use of the accelerometer to detect acceleration, orientation, and vibrations.

The primary additional options to consider for the output part of the user interface includes voice generation and haptic feedback.

**46. With regards to social media, what is the difference between "push" and "pull" approaches to interacting with customers?**

If the user must come to you to find out something, then you are using a pull-based approach. On the other hand, if you put the information out to the user, then you are using a push-based approach. When it comes to social media, you really need to use a combination of the approaches. For example, in Facebook if someone posts on your wall or sends you a request, Facebook will send you an email message to try an entice you back to the Facebook site. The act of posting to your site was a pull-based action, while the email message sent to you is a push-based action. In a nutshell, you want to focus on more of a push-based approach. You want your content to get to your customers in as an effective manner as possible. You don't want them to have to come looking for you. Encourage them to opt-in for update notifications to come to them in a form that they prefer. Some may prefer email notifications, while others may prefer you post to their Facebook or Twitter accounts. Also, be sure to include links to your social media sites on your home page. But, be sure not to overwhelm the customer. Not every customer wants to know every tidbit regarding the firm. Only give the customer what the customer wants. Remember, Krug's first principle: Don't Make Me Think! A corollary to this principle for social media would be: Don't Make Me Work! So, make it easy for the customer to find only what they want (or maybe what we want them to want).

**47. Why is it important to keep your social media sites synced?**

Be sure that your home page and your social media sites are all synced together so that when one is updated, the other sites "know" about the update. This will make your job of maintaining the different sites much easier and it will allow your customers to have a consistent experience across all sites. However, don't overdue this. It is obvious that different sites have different media and, potentially, different audiences. You aren't going to use Facebook in the same way you would use Twitter, YouTube, or a blog. Also, be sure to include crosslinks between the different sites. This will enable your customer to easily navigate through your different sites. For example in Figure 10-27, we see that Wiley has supported the ability to join email lists and the ability to link to Apple's iTunes store to download apps produced by Wiley on their home page. However, they do not provide links to either their Facebook or Twitter sites.

**48. How can you keep your customers engaged with your social media sites?**

You can include buttons that allow them to email the content to their closest "friends" or other followers in their own social network. You also should provide a means to gather feedback from your customers regarding your content. One way is to include the ability for customers to make and share comments regarding your content. Another way is to provide a voting or "like" mechanism to encourage the customer to become engaged with your site.

**49. Why do people play games?**

Typically, you play games because you enjoy the game. Other reasons include as a diversion and as a way to socialize with other game players.

**50. What is gamification?**

Gamification deals with applying gaming mechanics to non-gaming situations. Gamification has been used to redesign classrooms, to support learning, and, like games, it too has been used to increase customer and employee engagement

**51. What is occlusion? Why is it an issue when developing multi-dimensional information visualizations? Augmented reality systems? Virtual reality systems?**

Occlusion is when viewing data in 3D, some of the visualization may be covered up, hidden, by other parts of the visualization. It blocks information in the back.

Augmented and virtual reality using immersive technologies, is among the latest and exciting application areas being utilized to solve business problems. Where virtual reality (VR) technologies completely immerse the user into an artificial simulated digital environment, augmented reality (AR) technologies are used to augment or enhance the view of the real world. There are both opportunities and challenges with deploying both of these technologies.

**52. What is augmented reality?**

See prior answer.

**53. Name some of potential business applications of augmented reality.**

Game, sports, education, training.

**54. What is virtual reality?**

See answer in 51.

**55. Name some of potential business applications of virtual reality.**

Game, sports, education, training, entertainment.

**56. When developing a virtual reality system, what are some of the issues that need to be addresses?**

Obviously, designing effective and efficient VR applications is very difficult39. Again, the overall design process is similar to the general user interface design process described earlier. However, given the potential for VR to support business decision-making by combining gaming and information visualization technologies into a single seamless distributed environment and that the investment in specialized hardware and software is dropping, VR could provide large payoffs.

**57. What is a cognitive map?**

A cognitive map consists of not only spatial relationships, but also of auditory, sensory and emotional impressions. In games, a map is typically provided to help with change "is" to "are" understanding where one is in the virtual space and from where one has come. Like the immersion issue, wayfinding also raises issues related to individual psychological and cognitive differences.

**58. What are some of the multilingual issues that you may face when developing for a global audience?**

Global applications often have multilingual requirements, which means that they have to support users who speak different languages and write using non-English letters (e.g., those with accents, Cyrillic, Japanese). One of the most challenging aspects in designing global systems is getting a good translation of the original language messages into a new language. Words often have similar meanings but can convey subtly different meanings when they are translated, so it is important to use translators skilled in translating technical words

**59. How important is the proper use of color when developing web sites for a global audience? Give some examples of potential pitfalls that you could run into.**

The meaning associated with a color is totally culturally dependent. In fact, black and white isn’t necessarily black and white; they could be white and black. In most Western cultures, black is associated with death, mourning, and grief or with respect and formality. For example, in the US, we typically wear black to a funeral or you would expect to see religious leaders in black (think about the robes typically worn by a Catholic priest). While in many Eastern cultures, white is associated with death or the color of robes worn by religious leaders. In an example reported by Singh and Pereira, when senior citizens in the US and India were asked to “visualize the following statement: A lady dressed in white, in a place of worship,” the results that came back were as near to the opposite as one could get. In India, the lady would be a widow, while in the US she would be expected to be a bride.

Other colors that have meanings that are culturally driven include green, blue, red, yellow, and purple. In the US, red implies excitement, spice passion, sex, and even anger, in Mexico, it indicates religion, in the UK, authority, power, and government, in Scandinavian countries, strength, while in China, it means communism, joy, and good luck; quite a diversity of meanings. The color blue is associated with Holiness in Israel, cleanliness in Scandinavia, love and truth in India, loyalty in Germany, and in the US, it represents trust, justice, and "official" business. In Ireland, the color green signifies nationalism and Catholicism, while in the US, it denotes health, environmentalism, safety, greed, and envy. To say the least, green is a very confusing color for Americans. While in the Arab Middle East, green is a sign of Holiness, in

France, green represents criminality, and in Malaysia, it signifies danger and disease. The color yellow also has many culturally dependent meanings. In the US, it is associated with caution and cowardice, in Scandinavia, warmth, in Germany, envy, and in India, commerce. Finally, the color purple signifies, death, nobility, or the Church in Latin America, US, and Italy, respectively. Obviously, when building a web site for a global audience, colors must be chosen carefully. Otherwise, unintentional messages will be sent.

**60. Name the three cultural dimensions that are relevant to user interface design identified by Hall. Why are they relevant?**

Hall identified three dimensions that are directly relevant to user interface design: speed of messages, context, and time. The *speed of messages* dimension deals with how fast a member of a culture is expected to understand a message and how "deep" the content of a typical message will be in a culture. The deeper the message content, the longer it will take for a member of a culture to understand the message. For example, two different approaches to describe a historical event would be a news headline (fast and shallow) and a documentary (slow and deep). According to Hall, different cultures will have different expectations as to the content of and response to a message. This particular dimension has implications for the content of the message contained in the user interface. Krug's third design principle turns out to be culturally driven. For a Western audience, minimizing the number of words contained in a user interface makes sense. Westerners prefer to get to the point as fast as possible. However, this is not true for Eastern cultures. Consequently, for a firm like Amazon.com, providing detailed reviews and short excerpts from a book provides support for a slow and deep culture, while providing bullet point types of comments supports the fast and shallow culture. By providing both, Amazon.com addresses both needs.

The second dimension, *context*, deals with the level of implicit information that is used in the culture versus the information needing to be made explicit. In high context cultures, most information is known intrinsically and does not have to be made explicit. Therefore, the actual content of the message is fairly limited. However, in low context cultures, everything must be spelled out explicitly to avoid any ambiguity, and therefore the message will need to be very detailed. You will find this dimension causing problems when attempting to close a business deal. In most Western societies, the lawyers want everything spell out. In contrast, in most Eastern societies, it may, in fact, be considered insulting to have to spell everything out. From a web site design perspective Singh and Pereira point out that in a high context culture, focusing the design on aesthetics, politeness, humility will produce an effective web site, while in a low context culture, things such as the terms and conditions of a purchase, the "rank" of the product and firm, and the use of superlatives in describing the product and firm are critical attributes of a successful web site.

Hall's third dimension, *time*, addresses how a culture deals with many different things going on simultaneously. In a *polychronic time* culture, members of the culture will tend to do many things at the same time, but be easily distracted, and view time commitments as something that is very flexible. Whereas with *monochronic time* cultures, members of the culture will solve many things by focusing on one thing at a time, being single-minded, and consider time commitments as something that is set in stone. When designing for a polychronic culture, the liberal use of "pop-up" messages might be "fun" and engaging, while in a monochronic culture, "pop-up" messages simply annoy the user. In the past, northern hemisphere-based cultures have been monochronic and southern hemisphere-driven cultures have been polychronic. However, with the use of email interrupts and text messaging, this could change over time.

**61. Name the four cultural dimensions that are relevant to user interface design identified by Hofstede. Why are they relevant?**

The first dimension, power distance, addresses how the distribution of social power is dealt with in the culture. In cultures with a high power distance, members of the culture believe in the authority of the social hierarchy. In low power distance cultures, members of the culture believe that power should be more equally distributed. Consequently, in cultures with a high power distance, emphasis on the "greatness" of the leaders of the firm, the use of "proper titles" for members of the firm, and the posting of testimonials on behalf of the firm by "prominent" members of society is important. International awards won by the firm, its members, or its products should also be posted prominently on the website.

The second dimension, uncertainty avoidance, addresses to what degree a culture is comfortable with uncertainty. A culture with a high uncertainty avoidance will have members who will avoid taking risks, value tradition, and be much more comfortable in a rule-driven society. The higher the culture is on uncertainty avoidance, the more customer service will need to be provided, the more important "local" contacts need to be available, an emphasis on the firm's and product's history and tradition is provided on the web site, and in the case of software, the use of free trials and downloads is critical. In other words, you need to build trust and reduce perceived risk between the customer and the firm. This can be supported through things such as product seals of approval or the use of WebTrust and SysTrust certifications for the Web site. Furthermore, merely translating a web site from a low uncertainty avoidance culture to a high uncertainty avoidance culture will not be sufficient. You also need to point out relationships between the local culture and the firm's products.

The third dimension, individualism versus collectivism, is based on the level of emphasis the culture places on the individual or the collective, or group. In the West, individualism is rewarded. However, in the East, it is believed by focusing on optimizing the group, the individual will be most successful. In other words, it is the group that is the most important. In a collective society, presenting information on how the firm "gives back" to the community, supports "member" clubs, "loyalty" programs, and "chat" facilities, and provides links to "local" sites of interest are very important characteristics for a web site. In contrast, in an individualistic society, providing support for personalization of the user's experience with the web site, emphasizing the uniqueness of the products that the user is viewing, and emphasizing the privacy policy of the site are critical.

Hofstede's fourth dimension, masculinity versus femininity, does not mean how men and women are treated by the culture. But, instead this dimension addresses how well masculine and feminine characteristics are valued by the culture. For example, in a masculine culture characteristics such as being assertive, ambitious, aggressive, competitive are valued, while in a feminine culture, characteristics such as being encouraging, compassionate, thoughtful, gentle, cooperative are valued. In masculine cultures, a focus on the effectiveness of the firm's products is essential. Also, clearly separating male and female-oriented topics and placing them on different sections of a web site can be critical. According to Singh and Pereira, feminine cultures will value a focus on aesthetics and using more of a "soft-sell" approach where the focus is on more affective, intangible aspects of the firm, its members, and its products is more appropriate.

Obviously, operationalizing Hall's and Hofstede's dimensions for effective user interface design is not easy. However, given today's global markets, ignoring cultural issues in user interface design, whether it is for an internal system used only by employees of the firm or an external system that is used by customers, will most certainly cause a system to fail. This is especially true when you consider mobile and social media sites.

**62. What are some of the nonfunctional requirements that can influence the design of the human–computer interaction layer?**

Operational requirements, such as choice of hardware and software platforms, influence the design of the human–computer interaction layer. For example, something as simple as the number of buttons on a mouse (one, two, three, or more) changes the interaction that the user will experience. Other operational nonfunctional requirements that can influence the design of the human–computer interaction layer include system integration and portability. In these cases, a Web-based solution may be required, which can affect the design; not all features of a user interface can be implemented efficiently and effectively on the Web. This can require additional user-interface design. Obviously, the entire area of mobile computing can affect the success or failure of the system.

Performance requirements, over time, have become less of an issue for this layer. However, speed requirements are still paramount; especially with mobile computing. Most users do not care for hitting return or clicking the mouse and having to take a coffee break while they are waiting for the system to respond, so efficiency issues must be still addressed. Depending on the user-interface toolkit used, different user-interface components may be required. Furthermore, the interaction of the human–computer interaction layer with the other layers must be considered. For example, if the system response is slow, incorporating more-efficient data structures with the problem domain layer, including indexes in the tables with the data management layer, and/or replicating objects across the physical architecture layer could be required.

Security requirements affecting the human–computer interaction layer deal primarily with the access controls implemented to protect the objects from unauthorized access.

Most of these controls are enforced through the DBMS on the data management layer and the operating system on the physical architecture layer. However, the human–computer interaction layer design must include appropriate log-on controls and the possibility of encryption.

In addition to the international and cultural issues describe previously, unstated norms affect the cultural and political requirements that can affect the design of the human–computer interaction layer. Unstated norm requirements include having the date displayed in the appropriate format (MM/DD/YYYY versus DD/MM/YYYY). For a system to be truly useful in a global environment, the user interface must be customizable to address local cultural requirements.

**63. Why it is important to perform an interface evaluation before the system is built?**

An interface assessment is important before the system is built because we need to do as much as we can to improve the interface design prior to implementation. It is wasteful to wait until after implementation to evaluate the interface because it will be expensive to go back and modify the interface at that point.

**64. Compare and contrast the five types of interface evaluation.**

There are five common approaches to interface evaluation: heuristic evaluation, walkthrough evaluation, interactive evaluation, A/B testing, and formal usability testing.

1. A heuristic evaluation examines the interface by comparing it to a set of heuristics or principles for interface design.
2. An interface design walkthrough evaluation is a meeting conducted with the users who ultimately have to operate the system.
3. With an interactive evaluation, the users themselves actually work with the user interface prototype in a one-person session with members of the project team.
4. A/B testing is a form of interactive evaluation. Potential users are assigned to the different versions of the user interface. The effectiveness of the user interfaces are tested and statistically analyzed to determine which of the user interfaces should be further developed and deployed.
5. Formal usability testing is commonly done with commercial software products and products developed by large organizations that will be widely used through the organization.

**65. Under** what **conditions is heuristic evaluation justified?**

Heuristic evaluation is probably justified in situations where the interface is well understood. When there is little uncertainly about how the interface should function, then it is probably sufficient to just assess it internally by comparison to a checklist of design principles. It would be dangerous to use this technique (which does not involve users) if there was uncertainty about what should appear in the interface or how it should function.

# Chapter 11 Physical Architecture Layer Design End-of-Chapter Questions

1. **What are the four basic functions of any information system?**

   The four basic functions of any information system are:

   1. Data storage
   2. Data access logic
   3. Application logic
   4. Presentation logic

2. **What are the three primary hardware components of any physical architecture?**

   The three primary hardware components of any physical architecture are:

   1. Servers
   2. Client computers
   3. Network

3. **Name two examples of a server.**

   Servers are typically larger computers that are used to store software and hardware that can be accessed by anyone who has permission. Servers can come in several flavors: mainframes, minicomputers and microcomputers.

4. **Compare and contrast server-based architectures, client-based architectures and client-server- based architectures.**

   Although there are numerous ways in which the software components can be placed on the hardware components, there are three principal application architectures in use today: server-based architectures, client-based architectures and client–server architectures. The most common architecture is the client–server architecture.

   Server-based architecture: Being one of the very first computing architectures, this had the server performing all the four application functions.

   **Advantages:**
   - Simple architecture works well
   - Single point of control cause all messages flow through server
   - Application software is developed and stored on one computer and all the data are on the same computer.

**Disadvantages:**
- Server must process all messages
- Increase in applications results in slower response rate
- Server upgrades come in large increments and are expensive.

Client-based architecture: Clients generally are personal computers on a local area network and the server computer is on the same network.

**Advantages:**
- Simple architecture and works well

**Disadvantages:**
- All data on the server must travel to the client for processing, resulting in network overload and high power requirement on client computers.

Client-Server architecture: Most organizations today are moving to client–server architectures, which attempt to balance the processing between the client and the server by having both do some of the application functions. In these architectures, the client is responsible for the presentation logic while the server is responsible for the data access logic and data storage. The application logic may reside on either the client, the server, or be split between both.

**Advantages:**
- They are scalable
- They are able to support different types of clients and servers.
- For thin client server architectures that use Internet standards, it is simple to clearly separate the presentation logic, the application logic, and the data access logic and design each to be somewhat independent.
- Network stability

**Disadvantages:**
- Writing software is more complicated because client-server computing requires two applications: one on the client and the other on the server.
- Updating the network with a new version of the software is more complicated, too.
- Software updating is to be done in all servers and clients.

**5. What is the biggest problem with server-based computing?**

The server must process all messages. Therefore, as the volume of use grows, the systems may become overloaded and unable to keep up with demand.

6. **What is the biggest problem in client-based computing?**

   The fundamental problem in client-based networks is that all data on the server must travel to the client for processing.

7. **Describe the major benefits and limitations of thin client-server architectures.**

   1. For thin client server architectures that use Internet standards, it is simple to clearly separate the presentation logic, the application logic, and the data access logic and design each to be somewhat independent.
   2. Simple program statements are used to link parts of the interface to specific application logic modules that perform various functions.
   3. It is possible to change the application logic without changing the presentation logic or the data, which are stored in databases and accessed using SQL commands.

   The most important limitation in a thin-client server architecture is the complexity.

8. **Describe the major benefits and limitations of thick client-server architectures.**

   The major benefit of thick client-server architecture is that the network load is reduced as the presentation logic and application logic resides at the client end itself. The programming is made considerably easier. The major limitation is the increased cost of maintenance and support overheads.

9. **Describe the differences between two-tiered, three-tiered and n-tiered architectures.**

   Since the application has four components, these can be organized between two layers, three layers, or four (and conceivably more) layers. Generally the presentation logic will reside with the client and the database in a more centralized server; however application and data access logic will vary in terms of where they are located.

   <u>Two-tiered architecture</u>: A two-tiered architecture is one of the most common. In this case the server is responsible for the data and the client is responsible for the application and presentation. It is called two-tiered because uses only two sets of computers, clients and servers.

   <u>Three-tiered architecture:</u> A three-tiered architecture uses three sets of computers In this case, the software on the client computer is responsible for presentation logic, an application server(s) is responsible for the application logic, and a separate database server(s) is responsible for the data access logic and data storage.

N-tiered architecture: An n-tiered architecture uses more than three sets of computers. In this case, the client is responsible for presentation, a database server(s) is responsible for the data access logic and data storage, and the application logic is spread across two or more different sets of servers.

**10. Define *scalable*. Why is this term important to system developers?**

Scalability refers to the ability to increase or decrease the capacity of the computing infrastructure in response to changing capacity needs. This term is important to the system developers because they can plan the design of the system to handle multiple servers as the application usage and data storage needs grow.

**11. What six criteria are helpful to use when comparing the appropriateness of computing alternatives?**

The six important criteria to use are:

1. Cost of the infrastructure
2. Cost of development.
3. Ease of development
4. Interface capabilities
5. Control and security and
6. Scalability.

**12. Why should the project team consider the existing physical architecture in the organization when designing the physical architecture layer of the new system?**

The existing architecture will affect most of the criteria as regards a particular project. For example, a mainframe-oriented organization may have much less expensive and easier development if it uses server-oriented architecture. If it were to shift to client-server architecture, it would be likely to invest in infrastructure and staffing over and above the direct cost of the project.

**13. Name the three different types of clouds. How do they differ from each other?**

There are three different classifications of clouds: private, public, and hybrid. Private clouds are available only to employees of the firm, public clouds are available to the general public, and hybrid clouds combine the private and public cloud ideas together to form a single cloud. In some senses, all e-commerce sites could run in a hybrid cloud environment where the customer sales transaction portion of the system would need to be public while all other aspects would be private.

**14. What is meant by a service-oriented architecture?**

Web services basically support connections between different services to form service-oriented architectures. Basically, a service is a piece of software that supports some aspect of a business process. A service can be an implementation of part of a business process, it can be an implementation of an entire business process (for example, salesforce.com), or it can be object persistence support for the data management layer. These services can be either internal or external to the firm. Services can be combined to support business processes. If you recall, we suggest modeling business process with use cases, use case diagrams, and activity diagrams. A service-oriented architecture allows business processes to be supported by "plugging and playing" services together in a static and/or dynamic manner. Furthermore, some of the pluggable and playable services can be purchased out right or they can be billed to the firm based on their use; a sort of pay as you go model.

**15. Define virtualization. How does it relate to the cloud?**

Virtualization is the idea of treating any computing resource, regardless of where it is located, as if it is "in" the client machine. This idea evolved from virtual memory. Virtual memory was developed originally in the 1960s. Virtual memory allowed the user/programmer to act as if the amount of main memory in the computer was unlimited. This was done by swapping pages of main memory out to disk when the contents of the pages was not being used and swapping a page from disk back to main memory when it was needed. Before virtual memory was created, the programmer had to write code to perform the paging function for each application. Virtualization is simply the scaling up of this idea to all computing resources, not simply main memory. This includes treating a mainframe computer as if it is a set of virtual servers, each of which can be running different operating and/or application systems.

**16. What is the difference between IaaS, PaaS, and SaaS?**

The cloud can contain the firm's IT infrastructure, IT platform, and software. Infrastructure as a Service (IaaS) refers to the cloud providing the computing hardware to the firm as a remote service. The hardware typically includes the computing hardware that supports application servers, networking, and data storage. Amazon's EC2 (aws.amazon.com/ec2/) service is a good example of this. With Platform as a Service (PaaS), the cloud vendor not only provides hardware support to a customer, but also provides the customer with either package-based solutions, different services that can be combined to create a solution, or the development tools necessary to create custom solutions in the PaaS vendor's cloud. SalesForce.com is a good example of the vendor providing a package-based solution, Amazon's SimpleDB and Simple Query Service is a good example of different services being supported, and Google's App Engine is a good example of a cloud vendor providing good development tools. Like most things in IT, Software as a Service (SaaS) is not a new idea. SaaS has been around for over 30 years. In the 1970s, there were many "service bureaus" that supported timesharing of hardware and software to many different customers, i.e., they supported multi-tenancy. For example, ADP has supported payroll functions for many firms for a very long time. Today, SalesForce.com's CRM system is a good example of a SaaS cloud-based solution.

**17. What are the obstacles for provisioning the physical architecture layer with cloud technologies?**

The first obstacle is the mixed level of cloud performance. A second obstacle deals with the level of dependency that a customer's firm has on a cloud vendor. A third major obstacle to cloud adoption is the perceived level of security available in the cloud.

**18. What, if any, are the issues related to security in the cloud?**

A major obstacle to cloud adoption is the perceived level of security available in the cloud. Not only does a firm have to worry about security from the outside, when you consider multi-tenancy, the firm must seriously consider potential attacks from within their cloud from other cloud users. Furthermore, from a service availability perspective, a denial-of-service attack against another tenant within the cloud can cause performance degradation of the firm's systems. Finally, a firm must consider protecting themselves from the cloud vendor itself. It turns out, the cloud vendor is responsible only for physical security and firewalls. All application-level security tends to be the responsibility of the cloud customer. Obviously, security in the cloud is a very complex endeavor. Given the confidentiality and auditability requirements of Sarbanes-Oxley (SOX) and the Health and Human Services Health Insurance Portability and Accountability Act (HIPAA), security in the cloud becomes a major concern for a firm to move any of its confidential data, including email, to the cloud. In many ways, when using a cloud a firm is simply taking a "leap of faith" that the cloud is secure.

**19. What are SOX and HIPAA and how could they affect a firm's decision to adopt cloud technology?**

Security in the cloud is a very complex endeavor. Given the confidentiality and auditability requirements of Sarbanes-Oxley (SOX) and the Health and Human Services Health Insurance Portability and Accountability Act (HIPAA), security in the cloud becomes a major concern for a firm to move any of its confidential data, including email, to the cloud. In many ways, when using a cloud a firm is simply taking a "leap of faith" that the cloud is secure.

**20. What is meant by ubiquitous computing? How about the Internet of Things?**

Essentially, ubiquitous computing is the idea that computing takes place everywhere and in everything. With ubiquitous computing, computing becomes so engrained into everyday things, computing effectively disappears into the background. In other words, computing becomes so deeply rooted into everyday things that the things themselves seem to become magical. The Internet of Things (IoT) is the idea that, in addition to things having some form of computing capacity built into them, everyday things become connected via the Internet.

**21. What is an enchanted object? Give a set of examples of them.**

An enchanted object is an everyday object that has a very specialized processor embedded in it that augments the object such that the object seems to be magical. For example, an umbrella that, since there is a good chance of rain, lets you know that you should take it with you today, or a wallet that lets you know that you are reaching your monthly budget limits or that your account just received a deposit.

**22. What is e-waste?**

Old computers or other computing devices.

**23. What is the problem with backyard recycling of e-waste?**

Owing to "backyard recycling" techniques used in these locations, the toxic material contained in the e-waste shows up in the soil, water, and air. (It is funny to ask this question in a software design course!)

**24. What is meant by a green data center?**

Large data centers use as much electricity in a day as a small city. Consequently, given this level of power consumption, creating green data centers in the future will be crucial. There are a whole set of ways to create a green data center. One way is to pay very close attention to where the data center is to be located. Placing the data center in the shade of a mountain or tall building will reduce the cost of energy required.

**25. How do tablets, such as the iPad, enable the paperless office?**

The paperless office idea has been around for a very long time. However, up until now, the idea has been more fantasy than reality. Today, with the advent of multiuse tablets, such as Apple's iPad, the paperless office is becoming a reality. When considering the cloud and the apps available on the iPad, it is possible to not only create a paperless office, but also to have the paperless office effectively to be a portable office.

**26. What additional hardware- and software-associated costs may need to be included on the hardware and software application?**

Some examples include warrantees, licenses or site licenses, and training.

**27. Who is ultimately in charge of acquiring hardware and software for the project?**

Frequently, the organization has a Purchasing Department or other team that is charged with conducting large purchases. Sometimes they are in a position to get bargains due to scale and otherwise negotiate with vendors as well as efficiently process the related paperwork.

**28. What is a benchmark and why is it important?**

Depending on the overall cost and size of the project, one thing that should be seriously considered is the use of a benchmark. A benchmark is essentially a sample of programs that would be expected to run on the new physical architecture. Even though benchmarks can be expensive to create, they tend to provide a more realistic picture of how the proposed physical architecture layer will perform

**29. Why is Parkinson's Law relevant to the design of the physical architecture layer?**

From an IT perspective, Parkinson's Law implies that regardless of the users' real needs, their imagined needs will always fill up whatever capacity the system has. Consequently, it is imperative that the physical architecture layer design be based on the current and expected future architecture of the problem domain layer.

**30. What do you think are three common mistakes that novice analysts make in physical architecture design and hardware and software specification?**

The three common mistakes that novice analysts make in physical architecture design is in analyzing the non-functional requirements and how they are likely to change over time, performance requirements on how the business changes are to be expected to which the systems would need to adapt, and, data security requirements. These decisions would impact the hardware and software specification too.

**31. Describe the major nonfunctional requirements and how they influence physical architecture layer design.**

The nonfunctional requirements developed in early during analysis (see Chapter 5) play a key role in physical architecture layer design. These requirements are reexamined and refined into more detailed requirements that influence the system's architecture.

There are four primary types of nonfunctional requirements that can be important in designing the architecture:

1. Operational requirements: Specify the operating environment(s) in which system must perform and how those may change over time. That is:
   - Technical environment
   - System Integration
   - Portability
   - Maintainability
2. Performance requirements: Focus on performance issues such as response time, capacity and reliability.
   - Speed
   - Capacity
   - Availability
   - Reliability
3. Security requirements: Ability to protect the information system from disruption and data loss, whether caused by international act or a random event.
   - System value
   - Access control
   - Encryption
   - Authentication
   - Virus control
4. Cultural/political requirements: Are specific to the countries in which the system will be used.
   - Multilingual
   - Customization
   - Unstated norms
   - Legal

**32. Why is it useful to define the nonfunctional requirements in more detail even if the technical environment requirements dictate a specific architecture?**

In many cases, the technical environment requirements as driven by the business requirements may simply define the physical architecture layer. In this case, the choice is simple: business requirements dominate other considerations.

In the event that the technical environment requirements do not require the choice of a specific architecture, then the other nonfunctional requirements become important. Even in cases when the business requirements drive the architecture, it is still important to work through and refine the remaining nonfunctional requirements because they are important in later stages of the design and implementation phases.

**33. What does the network model communicate to the project team?**

The network model shows the complexity of the system and how the major components fit together.

**34. What are the differences between the top-level network model and the low-level network model?**

The high-level diagram shows an overview of the network. Developers get a good understanding of the geographic scope as well as some of the security and possibly global aspects of the project. The low-level network shows the detailed contents of the locations shown on the high-level.

**35. Are some non-functional requirements more important than others in influencing the architecture design and hardware and software specification?**

Some non-functional requirements, such as operational, performance, security, and cultural and political requirements are important in influencing the physical architecture design and hardware and software specifications. Among these operational, performance and security requirements are common and usually have specific standards that need to be met. Occasionally, cultural and political requirement become more important. This is because these requirements are specific to the countries in which the system will be used. Examples of non-functional requirements include:

- <u>Multilingual requirements</u>: That is the language in which the system needs to operate.
- <u>Customization Requirements</u>: Specification of what aspects of the system can be changed by local users
- <u>Unstated Norms</u>: Unstated assumptions about proper behavior that differ from country to country
- <u>Legal Requirements</u>: The laws and regulations that impose requirements on the system

**36. What do you think are the most important security issues for a system?**

Security is the ability to protect the information system from disruption and data loss, whether caused by an intentional act (e.g., a hacker or a terrorist attack) or a random event (e.g., disk failure, tornado). Security is primarily the responsibility of the operations group—the staff responsible for installing and operating security controls, such as firewalls, intrusion detection systems, and routine backup and recovery operations. Nonetheless, developers of new systems must ensure that the system's security requirements produce reasonable precautions to prevent problems; system developers are responsible for ensuing security within the information systems themselves.

Some of the most important security issues are:

- <u>Access Control Requirements</u>: This means limitations on who can access what data.
- <u>Encryption and Authentication</u>: Defines what data will be encrypted where and whether authentication will be needed for user access.
- <u>Virus Control Requirements</u>: Requirements to control the spread of virus.

**37. Describe the different approaches to access control.**

Access control requirements state who can access what data and what type of access is permitted: whether the individual can create, read, update and/or delete the data. There are several approaches to address access control. An access control list is associated with the asset for which you want to control access. One way is by capturing the access control lists in the pre-conditions of a method contract. Another approach would be to create a capabilities list with each user profile. Finally, we could combine the access control and capabilities list approaches and require that access is only granted to an IT asset if both the access control list and the capabilities list match.

**38. Why is it important to have end user license agreements examined?**

An organization should restrict the installation of any software until the organization's lawyers clear it. Otherwise, a user could be granting access to an outside organization to organizational data and possibly to intellectual property. In other words, we need to have the "fine print" of the agreement read by an expert to prevent the accidental loss of organizational assets.

# Chapter 12 Construction
# End-of-Chapter Questions

1. **Why is testing important?**

   The costs associated with finding and fixing a major software bug after a system has been implemented are very high. Therefore, thorough testing of the system is imperative. It is not enough to have written a program; it also needs to be tested completely to verify it performs as specified.

2. **How can different national or organizational cultures affect the management of an information systems development project?**

   Information systems development teams can be geographically dispersed and multicultural in their membership, which adds a new wrinkle in the management of developing a successful information system. Some issues to consider are language barriers, different holiday schedules, and different time zones.

3. **What is the primary role of systems analysts during the programming stage?**

   Systems analysts typically do not do the programming, so during the programming stage, they work on other tasks, such as preparing test plans and performing testing as programs are developed.

4. **In *The Mythical Man-Month*, Frederick Brooks argues that adding more programmers to a late project makes it later. Why?**

   Adding programmers does not shorten the project time, and can actually increase it due to the increased communication and coordination that must take place on the programming team. When the team members have to spend time coordinating their work with others, they are not spending time programming, and their productivity diminishes.

5. **When offshoring development, how could differences in Hall's context dimension of culture affect the contribution of a team member to the successful development of an information system? What about Hall's time or speed of messages dimensions?**

   From an information systems development perspective, *context* could influence the ability of a team member to see (or not see) potential creative solutions that are out of the box or affect a team member's ability (or inability) to understand the entire problem under consideration. Furthermore, given this dimension, the level of detail in direction could be varied between cultures.

   When managing programmers in a multicultural setting, Hall's *time* dimension must also be considered. In *monochromic time* cultures, deadlines are critical. This is probably why *timeboxing* has been relatively successful as a method to control projects (see Chapter 2). However, in a *polychromic time* culture, a *deadline* is nothing more than a suggestion. Obviously, when managing programmers, understanding how the culture considers time is very important to have both a successful product delivery and a successful development process.

   Hall's *speed of messages* and context dimensions could also affect the manner in which this could be addressed. Depending on the culture, too much detail could be insulting, but attempting to put this issue in to a contextual frame that is culturally sensitive is difficult.

6. **What are Hofstede's five dimensions of cultural differences? How could differences in them influence the effectiveness of an information systems development team?**

   Hofstede's *individualism* and *collectivism* dimension partially explains the results regarding plagiarism and cheating described above. Given the importance that intellectual property plays in IT, this potentially could be a real problem when offshoring development to a collectivist culture.

   Hofstede's other previously mentioned dimensions are *power distance*, *uncertainty avoidance*, and *masculinity versus femininity*. Managing programmers in a culture with a high power distance value will be different than with a culture with a low power distance. For example, in the US, programmers see themselves as equals to their managers. In fact, in some firms, the president of the firm will be "coding" solutions alongside of a brand new hire. This somewhat explains the growing popularity of Agile methods (see Chapter 1). In comparison, the president of the firm would never stoop down to perform the same tasks as a new hire. It would be insulting to the president and embarrassing to the new hire. With regards to uncertainty avoidance, the choice of systems development approach could be affected. In a culture that prefers everything to be neat and ordered, a system development methodology that is very rule-driven would be beneficial. Furthermore, development team member professional certification and team/firm ISO or CMMI certifications would lend credibility to the team, whereas, in a culture that willingly takes on risk, certifications may not increase the perceived standing of the development team. When managing programmers in a masculine culture, it is critical to provide recognition to the top-performing members of the development team and also to recognize the top-performing teams. On the other hand, when considering a feminine culture, it is more important to ensure that the workplace is a supportive, non-competitive, and nurturing environment.

Finally, Hofstede identified a fifth dimension, *long*- versus *short-term orientation*, which deals with how the culture views the past and the future. In a long-term focused culture, team development and a deep relationship with a client is very important, while in a culture that emphasizes the short-term, delivering a high-quality product on time is all that really matters.

**7. What are the common language or languages used today in information systems development?**

Modeling
- UML

Programming
- Java
- C/C++
- Visual Basic
- PHP
- ASP

Database
- SQL

**8. Compare and contrast user documentation and system documentation.**

System documentation is created for the system developers, in order to document what was done during the development process, and to help them maintain the system after its installation. User documentation is designed specifically for the system users to help them work successfully with the system.

**9. Why is online documentation becoming more important?**

Online documentation is popular for several reasons.

- It is often easier to find information using the search index.
- It is possible to present help information in multiple formats, increasing its utility to the end users.
- Presenting information while interacting with the computer permits novel ways of presenting the information to be employed.
- The cost of distributing online documentation is considerably less than printing and distributing paper manuals.

**10. What are the primary disadvantages of online documentation?**

One problem is simply that many users are unfamiliar with seeking information online. We have all been exposed to paper manuals, and they are more familiar to many users. Also, online documentation is not portable. It is not possible to take an online help system home and review it before using a system as one could with a procedure manual or tutorial.

**11. Compare and contrast reference documents, procedures manuals, and tutorials.**

Reference documents are the help system that explains how to perform a specific task or function. Procedures manuals describe how to accomplish a particular business task with the system, often involving several steps. Tutorials are training systems that teach the user how to use the system.

**12. What are five types of documentation navigation controls?**

A table of contents organizes the information in a logical form. An index provides access to the help topics through keywords listed in the index. Test search allows the user to search for any topics that match the text entered by the user. An intelligent agent can be used to assist the user in the search. Links can be established that permit the user to jump between related topics.

13. **What are the commonly used sources of documentation topics? Which is the most important? Why?**

- One source is the set of commands and menus in the user interface.
- A second source is the use cases for the system that outline commonly performed tasks.
- A third source is the important terms associated with the system.

The most important source is probably the use cases for the system, since these define how to perform needed functions with the system.

14. **What are the commonly used sources of documentation navigation controls? Which is the most important? Why?**

The table of contents is developed directly from the logical structure of the documentation topics. Index terms can come from the user interface commands and functions, the major concepts (entities) of the system, the business tasks performed by the system user, and synonyms or alternate terms for the above.

15. **What is the purpose of testing?**

It is to find errors so that they can be corrected in a timely and cost-effective manner. The earlier errors are found in the process, the less it costs to correct them

16. **Describe how object-orientation impacts testing?**

Although it is tempting to think that reuse of components and other object oriented attributes makes testing less important, the opposite is true. More thorough testing is more difficult and, thus, more critical. Testing must be done systematically and the results documented so that the project team knows what has and has not been tested.

17. **Compare and contrast the terms test, test plan, and test cases.**

A test is a particular aspect of the system that needs to be evaluated. The test plan outlines all of the tests that need to be performed on the system. Test cases are the sample data that will be run through or entered into the system. A test will state its objective, list the test cases that will check the desired feature or function, and define the expected results. Actual test results will also be recorded on the test.

**18. What is a stub and why is it used in testing?**

To test modular programs it is not necessary to code the control module plus all the subordinate modules prior to testing anything. In fact, the control module should be tested before all the other modules are complete. In this situation, the subordinate modules are written as stubs, meaning that they are just program fragments that indicate that they have been successfully entered from the control module. If the control modules successfully access all its subordinate stub modules, then its structure has been tested. The subordinate module code can be added as it becomes available.

**19. What is the primary goal of unit testing?**

A unit test checks a single program or program module to ensure that it performs the function designated in the program specification.

**20. How are the test cases developed for unit tests?**

Each of the prior developed diagrams can serve as a basis for identifying tests. All invariants on the CRC cards, the state-chart diagrams, all pre and post conditions, and all contracts should be investigated.

**21. Compare and contrast black box testing and white-box testing.**

White box testing is based on the method specification associated with each class. In object-oriented development, this sort of testing is less important than black box testing where tests are developed directly from the specifications of each item in each class.

**22. What are the different types of class tests?**

Class tests, also called unit tests, include both black box and white box testing.

**23. What is the primary goal of integration testing?**

Integration testing evaluates whether a set of classes work together successfully. The span of an integration test is broader than unit testing, but not as broad as system testing.

**24. How are the test cases developed for integration tests?**

Generally, we start with the integration of the classes or collaboration used to support the highest priority use case.

**25. Describe the yo-yo problem. Why does it make integration testing difficult?**

The yo-yo problem occurs when the analyst or designer must bounce up and down through the inheritance graph to understand the control flow through the methods being executed. In most cases, this is caused by a rather deep inheritance graph; that is, the subclass has many superclasses above it in the inheritance graph. The yo-yo problem becomes even more of a nightmare in testing object-oriented systems when inheritance conflicts exist and when multiple inheritance is used

**26. What is the primary goal of system testing?**

System testing evaluates the entire system, not only to verify that it runs without error or breakdown particularly under the stress of heavy workload and potential security violation, but also that it fulfills the business requirements of the system. It also tests the system documentation.

**27. How are test cases developed for system tests?**

Test cases come from the requirements for the system as outlined in the system design documents, plus the unit tests and integration tests. Also, based on the system's business requirements, the user interface will be tested, security will be tested, performance under heavy processing volumes will be tested, and documentation will be tested.

**28. What is the primary goal of acceptance testing?**

These tests are performed with the system users to confirm that the system is complete, that it meets the needs of the business area, and that it is acceptable to the users.

**29. How are the test cases developed for acceptance tests?**

The users develop the test data for the initial acceptance testing. Their test cases will reflect their expected use of the system. In alpha testing, the data is made up, but in beta testing it uses selected but real data.

**30. Compare and contrast alpha testing and beta testing.**

The term 'alpha testing' is the preliminary stage of acceptance testing in which the users work with and test the system using dummy' data. In the 'beta testing' stage of acceptance testing, the users begin to use the system with 'live' data, but are closely monitored for errors and problems.

# Chapter 13 Installations and Operations End-of-Chapter Questions

1. **What are the three basic steps in managing organizational change?**

   According to Kurt Lewin change is a three-step process:

   1. Unfreeze - preparing people and the organization to break out of their current way of doing things
   2. Move - transition from the old to the new way of doing things
   3. Refreeze - establish the new system as the way things are done.

2. **What are the cultural issues of which developers should be aware?**

   The major cultural issues of which developers should be aware are power distance, uncertainty avoidance, individualism versus collectivism, masculinity versus femininity, and long- versus short-term orientation.

3. **What are the major components of a migration plan?**

   One part of the migration plan is the technical plan, which discusses how the new technology will be implemented in the organization. The other major part of the migration plan is the organizational plan, which helps users adjust to and adopt the new system.

4. **Compare and contrast direct conversion and parallel conversion.**

   Both conversion strategies focus on the way in which users are switched over to the new system. Direct conversion is an abrupt change: the old system is 'unplugged' and the new system is turned on. The users have no choice but to work with the new system. In the parallel conversion strategy, the new system is turned on, but the old system is used simultaneously for a time. Clearly, the direct strategy is more risky because there is no fallback if significant system bugs are discovered.

   With parallel conversion, the old system will still be available if there is a major bug discovered in the new system. However, the cost and extra work of using both systems simultaneously must be considered.

5. **Compare and contrast pilot conversion, phased conversion, and simultaneous conversion.**

   These terms refer to how the organizational locations are transitioned from the old system to the new system. In a pilot conversion, one location or part of the organization is converted first. The system is installed at that location and is used for a time until it appears to be stable (initial bugs discovered and fixed). Then the system is installed at the other locations. The pilot location serves as a shakedown site for the system. In the phased conversion approach, the system is installed sequentially at different locations. Gradually, the conversion process spreads across all locations until all have been converted. Simultaneous conversion means that all locations are converted at the same time. With this strategy, there is no period in which some locations are using the old system and other locations are using the new system.

6. **Compare and contrast modular conversion and whole system conversion.**

   Whole system conversion means that the entire system is installed at one time. This is the most common circumstance. Modular conversion, in which the various modules or components of the system are installed gradually over time, is less common, but can be used if the system is large and complex and has been designed as a set of independent system modules.

7. **Explain the trade-offs among selecting between the types of conversion in questions 4, 5, and 6 above.**

   - Direct conversion will be the quickest and lowest cost conversion style. Parallel conversion reduces risk, but costs much more and takes more time to complete.
   - Pilot conversion is the least risky way to convert organizational location. By debugging the system at the pilot location, the installation at the other locations should go smoothly. Phased conversion adds a little more risk in that time is not devoted to debugging the system totally before moving on to other sites. Phased conversion takes longer, however, than pilot conversions. The simultaneous approach will be the highest risk and most expensive approach of the three, but takes the least amount of time.
   - Whole system conversion is most risky, especially if the system is large and complex, but completes the conversion in the shortest amount of time. Modular conversion reduces risk somewhat because the system is implemented gradually. It adds cost and time to the process, however.

**8. What are the three key roles in any change management initiative?**

The project sponsor is the person or organizational unit that wants the change. The change agent is the person or group who leads the change effort. The potential adopter is the person or group who must actually do the changing.

**9. Why do people resist change? Explain the basic model for understanding why people accept and resist change.**

Resistance to change is a typical response to an externally imposed need to change. In determining their own response to a need to change, people will evaluate (implicitly or explicitly) the potential value of the new system (expected costs versus expected benefits), and will also assess their perceptions of the expected costs and benefits of transitioning from the status quo to the new situation. Only if the perceived benefits of the change and the perceived benefits of the transition outweigh the perceived costs will people's resistance to change be low.

**10. What are the three major elements of management policies that must be considered when implementing a new system?**

1. One element is the organization's standard operating procedures. These procedures define proper behavior, and communicate behavioral norms to the employees.
2. The second element is defining measurements and rewards, which explains what is important to the organization (worth measuring) and how desired behavior will be reinforced with rewards.
3. The third element is resource allocation. By allocating resources in certain ways, management can send both actual and symbolic messages to the organization.

**11. Compare and contrast an information change management strategy with a political change management strategy. Is one "better" than the other?**

Both of these change management strategies are focused on convincing the target adopters of the change that the benefits of the change outweigh the costs of the change. An informational approach may work when those change benefits really do exceed the change costs, but the target adopters may not have enough information to realize it. The informational strategy seeks to give them that information and encourage adoption because they now perceive the advantages of the new system. With the political change management strategy, organizational power is used to motivate change, not information. The change may not actually provide a direct benefit to the target adopters, but it may benefit the organization as a whole. In this case, political power may be employed to lead the target adopters to the conclusion that they should comply with the change (or pay a penalty).

**12. Explain the three categories of adopters you are likely to encounter in any change management initiative.**

There will usually be a small group (20-30 percent) who are ready adopters. These individuals quickly adopt the change. Another group of about the same size (20-30 percent) will be resistant adopters. They will refuse to accept the change and may actively fight against it. The remaining larger group is the reluctant adopters. These people will go whichever way the organizational wind is blowing.

**13. How should you design what items to include in your training plan?**

- Focus on helping the users to accomplish their jobs, not just on how to use the system.
- Focus on the system as it exists in its organizational context of getting day to day business done.
- Keep the emphasis on what the user needs to do; don't get caught up in what the system can do. Referring to the use cases can be helpful here, since if we design our training to cover the use cases, we should be able to make the users very comfortable with the system for their routine tasks.

**14. Compare and contrast three basic approaches to training.**

1. **Classroom training**: This is the traditional approach to training in which a number of people are trained simultaneously with one instructor. Classroom training is moderately effective. Its costs are moderate, and it is fairly effective, especially since it is the method that many people are most familiar with.
2. **One-on-one training**: This puts one trainer with one student at a time. One-on-one training is very effective, but is expensive to deliver and reaches only a few people
3. **Computer-based training**: This uses a CD or Web-based training program delivered to students as needed. Computer-based training is very costly to develop, but is inexpensive to disseminate after development. It can reach a large number of people, but its effectiveness can be limited.

**15. What is the role of the operations group in the system development?**

The operations group takes the system that the development team has created and makes it work for the organization on a daily basis.

**16. Compare and contrast two major ways of providing system support.**

System support is focused on helping users understand how to perform tasks and answering user questions. This can be accomplished with online support such as reference documentation and help screens, and also with special web sites that answer frequently asked questions. The goal here is to enable the user to find the answer to his/her question without talking to a human. When human help is needed, it is usually provided through a help desk. Help desks can provide human response to questions for the entire organization.

**17. How is a problem report different from a change request?**

A problem report documents a problem that has been encountered with the system that cannot be immediately resolved. The problem report will usually be passed to an application specialist, who will attempt to clear up the problem. If he/she cannot resolve the problem, then it is likely that a system bug has been encountered. At this point, the problem report becomes a change request, which is used to inform the system maintenance group that an un-resolvable problem has been discovered that probably requires a system fix.

**18. What are the major sources of change requests?**

- The most common source of change requests is the problem reports from the operations group that identify bugs encountered in the system.
- The second source is the system users, who submit ideas for minor changes and enhancements to the system.
- Third, other systems development project may have an effect on a system, and change requests may result in order to integrate the systems.
- The fourth source is senior management, who may trigger change requests in order to tailor the system to the organization's business strategy.

**19. Why is project assessment important?**

Project assessment is important because it enables the team and the IS organization to learn from its experience, and hopefully improve future systems development projects with the benefit of that experience.

**20. How is project team review different from system review?**

The project team review summarizes performance by the team members. Its purpose is to help the team members better understand what was done well and what needs improvement. The system review looks back at the projected costs and benefits for the system, and assesses whether these costs and benefits have been achieved following implementation. Estimates of future project costs and benefits can be improved if the accuracy of each project's estimates is evaluated.

**21. Why is the ideas of psychological safety, blameless post-mortems, and a just culture important when performing a project team review?**

By explicitly identifying mistakes and understanding their causes, project team members will, it is hoped, be better prepared for the next time they encounter a similar situation—and less likely to repeat the same mistakes. This is especially true when something fails, e.g., the team misses a deadline, the system fails tests, or the system blows up while in production. This type of review has become known as blameless post-mortems.

**22. What do you think are three common mistakes that novice analysts make in migrating from the AS-Is to the To-Be system?**

There are many possible answers to this question. Three of the more probably would include:

1. Underestimating the complexity and difficulty of gaining cooperation from project team, users, and other stakeholders.
2. Underestimating the time it takes to deal with the minute detail of each method, message, attribute, data type, and keep all the details aligned.
3. Underestimating the number of errors and difficulties that will appear when moving from the test to the user environment.

**23. Some experts argue that change management is more important than any other part of system development. Do you agree or not? Explain.**

Those who agree will point out that without appropriate change management, there is significant risk of employees not using or inappropriately using an expensive new tool (and thus not gaining the benefit from the development). Those who disagree will point out that great change management will not overcome horrible technical difficulties, such as a weak architecture or incompatible platforms. Though change management is important, it cannot be judged more important than the other constraining factors.

**24. In our experience, change management planning often receives less attention than conversion planning. Why do you think this happens?**

Conversion planning is an obvious and concrete activity needed to move a system from testing to production. Change management is often subtle, requires some recognition of the importance of interpersonal issues, and, particularly with managers that are quite authoritarian, may be viewed as a waste of time since employees should just do as instructed (in their view).

# Appendix 1 Basic Characteristics of Object-Oriented Systems End-of-Chapter Questions

**1. What is the difference between classes and objects?**

A class is the general template we use to define and create specific instances, or objects. Every object is associated with a class. An *object* is an instantiation of a class, which means an object has state, but not a class.

**2. What are methods and messages?**

Methods implement an object's behavior. A method is nothing more than an action that an object can perform. A message is essentially a function or procedure call from one object to another object.

**3. Why are encapsulation and information hiding important characteristics of object-oriented systems?**

In object-oriented systems, combining encapsulation with the information-hiding principle suggests that the information-hiding principle be applied to objects instead of merely applying it to functions or processes. As such, objects are treated like black boxes. The fact that we can use an object by calling methods is the key to reusability because it shields the internal workings of the object from changes in the outside system, and it keeps the system from being affected when changes are made to an object.

**4. What is meant by polymorphism when applied to object-oriented systems?**

Polymorphism means that the same message can be interpreted differently by different classes of objects. For example, inserting a patient means something different than inserting an appointment.

**5. Compare and contrast dynamic and static binding.**

Dynamic, or late, binding is a technique that delays typing the object until run-time. As such, the specific method that is actually called is not chosen by the object-oriented system until the system is running. This is in contrast to static binding. In a statically bound system, the type of object is determined at compile time. Therefore, the developer has to choose which method should be called instead of allowing the system to do it.

# Appendix 9-1 Optimizing RDBMS-Based Object Storage End-of-Chapter Questions

1. **Name three ways null values can be interpreted in a relational database. Why is this problematic?**

   The presence of null values suggests that space is being wasted. Null values also allow more room for error and increase the likelihood that problems will arise with the integrity of the data. Nulls threaten data integrity because they are difficult to interpret.

2. **What are the two dimensions in which to optimize a relational database?**

   The two dimensions in which to optimize a relational database are for storage efficiency and for speed of access

3. **What is the purpose of normalization?**

   Normalization is a process that optimizes relational data storage for storage efficiency and for minimization of update anomalies. The rules of normalization help assure that the data is stored as efficiently as possible.

4. **How does a model meet the requirements of third normal form?**

   First, the model must be in 1st normal form (all repeating fields or groups of fields have been removed to separate tables. Second, the model must be in 2nd normal form with all partial dependencies removed. Third normal form then requires that all transitive dependencies are removed (i.e., that no fields are dependent on other, non-primary key fields).

5. **Describe three situations that can be good candidates for denormalization.**

   Denormalization is performed to speed up data access. Redundancy is added back into tables in order to reduce the number of joins that are required to produce the desired information. In a normalized Order table, the customer name will not be included; however it may be added back in to the Order table to improve processing speed. This represents a situation in which some parent entity attributes are included in the child entity. Similarly, a lookup table of zip codes and states may be set up in the normalized data model, but could be added back in to the physical model design. Another situation is where a table of product codes lists the description and price. These may also be added back into the physical model to improve application performance. Lookup tables are common candidates for denormalization. Also, 1:1 relationships may be good candidates for denormalization, since the information may be accessed together frequently. Finally, storing parent and child relationships at the physical level can be more efficient with denormalized storage.

6. **Describe several techniques that can improve performance of a database.**

   Denormalization adds selected fields back to tables in a data model. This adds a little redundancy, but improves the data access speed. Clustering involves physically placing records together so that like records are stored close to each other. Indexing creates small, quickly searchable tables that contain values from the table and indicate where in the table those values can be found. Finally, proper estimation of the data set size is important to assure that adequate hardware is obtained for the system

7. **What is the difference between interfile and intrafile clustering? Why are they used?**

   Interfile clustering physically orders records within a table in some meaningful way, such as by primary key value. Interfile clustering identifies records for separate tables that are typically retrieved together and physically stores them together.

8. **What is an index and how can it improve the performance of a system?**

   An index is a small, quickly searchable table that contains values from the table and indicates where in the table those values can be found. System performance is improved with an index because it is no longer necessary to search the entire table for the desired values. The small index table can be quickly searched to reveal exactly where the desired values are stored.

**9. Describe what should be considered when estimating the size of a database.**

The size of the database will be based on the amount of raw data expected, the growth rate of raw data that is expected, and the overhead requirements of the DBMS.

**10. Why is it important to understand the initial and projected size of a database during design?**

The design team needs to be sure that the hardware that is specified for the system is adequate to support the size of the database. If inadequate hardware is chosen, the performance of the system will be poor regardless of the 'tuning' techniques that are applied.