Chapter 3:

Information Systems Acquisition, Development and Implementation

Overview

Part A: Information Systems Acquisition and Development

Part B: Information Systems Implementation

Case Study

Case Study
Answers to Case Study Questions

OVERVIEW

This chapter on information systems acquisition, development and implementation provides an overview of key processes and methodologies used by organizations when creating and changing application systems and infrastructure components.

This domain represents 12 percent of the CISA examination (approximately 18 questions).

DOMAIN 3 EXAM CONTENT OUTLINE

Part A: Information Systems Acquisition and Development

1. Project Governance and Management
2. Business Case and Feasibility Analysis
3. System Development Methodologies
4. Control Identification and Design

Part B: Information System Implementation

1. Testing Methodologies
2. Configuration and Release Management
3. System Migration, Infrastructure Deployment and Data Conversion
4. Post-implementation Review

LEARNING OBJECTIVES/TASK STATEMENTS

Within this domain, an IS auditor should be able to:

- Evaluate whether the business case for proposed changes to information systems meet business objectives. (T14)
- Evaluate the organization's project management policies and practices. (T16)
- Evaluate controls at all stages of the information systems development life cycle. (T17)

- Evaluate the readiness of information systems for implementation and migration into production. (T18)
- Conduct post-implementation review of systems to determine whether project deliverables, controls and requirements are met. (T19)
- Evaluate change, configuration, release, and patch management policies and practices. (T27)

SUGGESTED RESOURCES FOR FURTHER STUDY

ISACA, *ISACA IS Audit Basics*, *www.isaca.org/Knowledge-Center/ITAF-IS-Assurance-Audit-/Pages/is-audit-basics.aspx*

ISACA, *White papers*, *www.isaca.org/Knowledge-Center/Research/Pages/White-Papers.aspx*

ISACA, *COBIT Focus* articles, *www.isaca.org/COBIT/focus/Pages/FocusHome.aspx*

Pressmann, Roger S.; *Software engineering: A practitioner's approac*h, McGraw Hill, USA, 2010

Singleton, Tommie W.; "Auditing Applications, Part 1," *ISACA Journal*, vol. 3, 2012, *https://www.isaca.org/Journal/archives/2012/Volume-3/Pages/Auditing-Applications-Part-1.aspx*

Singleton, Tommie W.; "Auditing Applications, Part 2," *ISACA Journal*, vol. 4, 2012, *https://www.isaca.org/Journal/archives/2012/Volume-4/Pages/Auditing-Applications-Part-2.aspx*

Singleton, Tommie W.; "Auditing IT Risk Associated with Change Management and Application Development," *ISACA Journal*, vol. 5, 2011, *https://www.isaca.org/Journal/archives/2011/Volume-5/Pages/Auditing-IT-Risk-Associated-With-Change-Management-and-Application-Development.aspx*

Singleton, Tommie W.; "Testing Controls Associated With Data Transfers,"

ISACA Journal, vol. 2, 2012, *https://www.isaca.org/Journal/archives/2012/Volume-2/Pages/Testing-Controls-Associated-With-Data-Transfers.aspx*

SELF-ASSESSMENT QUESTIONS

CISA self-assessment questions support the content in this manual and provide an understanding of the type and structure of questions that typically appear on the exam. Often, a question will require the candidate to choose the **MOST** likely or **BEST** answer among the options provided. Please note that these questions are not actual or retired exam items. Please see the section "About This Manual" at the beginning of this manual for more guidance regarding practice questions.

3-1 To assist in testing an essential banking system being acquired, an organization has provided the vendor with sensitive data from its existing production system. An IS auditor's **PRIMARY** concern is that the data should be:

A. sanitized.
B. complete.
C. representative.
D. current.

3-2 Which of the following is the **PRIMARY** purpose for conducting parallel testing?

A. To determine whether the system is cost-effective
B. To enable comprehensive unit and system testing
C. To highlight errors in the program interfaces with files
D. To ensure the new system meets user requirements

3-3 When conducting a review of business process reengineering, an IS auditor found that an important preventive control had been removed.

In this case, the IS auditor should:

A. inform management of the finding and determine whether management is willing to accept the potential material risk of not having that preventive control.
B. determine if a detective control has replaced the preventive control during the process, and if it has not, report the removal of the preventive control.
C. recommend that this and all control procedures that existed before the process was reengineered be included in the new process.
D. develop a continuous audit approach to monitor the effects of the removal of the preventive control.

3-4 Which of the following data validation edits is effective in detecting transposition and transcription errors?

A. Range check
B. Check digit
C. Validity check
D. Duplicate check

3-5 Which of the following weaknesses would be considered the **MOST** serious in enterprise resource planning software used by a financial organization?

A. Access controls have not been reviewed.
B. Limited documentation is available.
C. Two-year-old backup tapes have not been replaced.
D. Database backups are performed once a day.

3-6 When auditing the requirements phase of a software acquisition, an IS auditor should:

A. assess the reasonability of the project timetable.

B. assess the vendor's proposed quality processes.
C. ensure that the best software package is acquired.
D. review the completeness of the specifications.

3-7 An organization decides to purchase a software package instead of developing it. In such a case, the design and development phases of a traditional system development life cycle would be replaced with:

A. selection and configuration phases
B. feasibility and requirements phases
C. implementation and testing phases
D. nothing, as replacement is not required.

3-8 User specifications for a software development project using the traditional (waterfall) system development life cycle methodology have not been met. An IS auditor looking for a cause should look in which of the following areas?

A. Quality assurance
B. Requirements
C. Development
D. User training

3-9 When introducing thin client architecture, which of the following types of risk regarding servers is significantly increased?

A. Integrity
B. Concurrency
C. Confidentiality
D. Availability

3-10 Which of the following procedures should be implemented to help ensure the completeness of inbound transactions via electronic data

interchange (EDI)?

A. Segment counts built into the transaction set trailer
B. A log of the number of messages received, periodically verified with the transaction originator
C. An electronic audit trail for accountability and tracking
D. Matching acknowledgment transactions received to the log of EDI messages sent

ANSWERS TO SELF-ASSESSMENT QUESTIONS

3-1 **A. Test data should be sanitized to prevent sensitive data from leaking to unauthorized persons.**

B. Although it is important that the data set be complete, the primary concern is that test data should be sanitized to prevent sensitive data from leaking to unauthorized persons.

C. Although it is important to encompass a representation of the transactional data, the primary concern is that test data should be sanitized to prevent sensitive data from leaking to unauthorized persons.

D. Although it is important that the data set represent current data being processed, the primary concern is that test data should be sanitized to prevent sensitive data from leaking to unauthorized persons.

3-2 A. Parallel testing may show that the old system is more cost-effective than the new system, but this is not the primary reason.

B. Unit and system testing are completed before parallel testing.

C. Program interfaces with files are tested for errors during system testing.

D. The purpose of parallel testing is to ensure that the implementation of a new system will meet user requirements.

3-3 **A. Those in management should be informed immediately to determine whether they are willing to accept the potential material risk of not having that preventive control in place.**

B. The existence of a detective control instead of a preventive control usually increases the risk that a material problem may occur.

C. Often during business process reengineering, many nonvalue-added controls will be eliminated. This is good, unless they increase the business and financial risk.

D. An IS auditor may wish to monitor or recommend that management monitor the new process, but this should be done only after management has been informed and accepts the risk of not having the preventive control in place.

3-4 A. A range check is checking data that match a predetermined range of values.

B. A check digit is a numeric value that is calculated mathematically and is appended to data to ensure that the original data have not been altered (e.g., an incorrect, but valid, value substituted for the original). This control is effective in detecting transposition and transcription errors.

C. An availability check is programmed checking of the data validity in accordance with predetermined criteria.

D. In a duplicate check, new or fresh transactions are matched to those previously entered to ensure that they are not already in the system.

3-5 **A. A lack of review of access controls in a financial organization could have serious consequences given the types of data and assets that could be accessed.**

B. A lack of documentation may not be as serious as not having properly reviewed access controls.

C. It may not even be possible to retrieve data from two-year-old backup tapes.

D. It may be acceptable to the business to perform database backups once a day, depending on the volume of transactions.

3-6 A. A project timetable normally would not be found in a requirements document.

B. Assessing the vendor's quality processes would come after the requirements have been completed.

C. The decision to purchase a package from a vendor would come after the requirements have been completed.

D. The purpose of the requirements phase is to specify the functionality of the proposed system; therefore, an IS auditor would concentrate on the completeness of the specifications.

3-7 **A. With a purchased package, the design and development phases of the traditional life cycle have become replaceable with selection and configuration phases. A request for proposal from the supplier of packaged systems is called for and evaluated against predefined criteria for selection, before a decision is made to purchase the software. Thereafter, it is configured to meet the organization's requirement.**

B. The other phases of the system development life cycle (SDLC) such as feasibility study, requirements definition, implementation and post-implementation remain unaltered.

C. The other phases of the SDLC such as feasibility study, requirements definition, implementation and post-implementation remain unaltered.

D. In this scenario, the design and development phases of the traditional life cycle have become replaceable with selection and configuration phases.

3-8 A. Quality assurance has its focus on formal aspects of software development such as adhering to coding standards or a specific development methodology.

B. To fail at user specifications implies that requirements engineering has been done to describe the users' demands. Otherwise, there would not be a baseline of specifications to check against.

C. **Obviously, project management has failed to either set up or verify controls that provide for software or software modules under development that adhere to those specifications made by the users.**

D. A failure to meet user specifications might show up during user training or acceptance testing but is not the cause.

3-9 A. Because the other elements do not need to change, the integrity risk is not increased.

B. Because the other elements do not need to change; the concurrency risk is not increased.

C. Because the other elements do not need to change, the confidentiality risk is not increased.

D. The main change when using thin client architecture is making the servers critical to the operation; therefore, the probability that one of them fails is increased and, as a result, the availability risk is increased.

3-10 **A. Control totals built into the trailer record of each segment is the only option that will ensure all individual transactions sent are received completely.**

B. A log of the number of messages received provides supporting evidence, but their findings are either incomplete or not timely.

C. An electronic audit trail provides supporting evidence, but their findings are either incomplete or not timely.

D. Matching acknowledgment transactions received to the log of electronic data interchange (EDI) messages sent provides supporting evidence, but their findings are either incomplete or not timely.

PART A: INFORMATION SYSTEMS ACQUISITION AND DEVELOPMENT

3.0 INTRODUCTION

For an IS auditor to provide assurance that an organization's objectives are being met by the management practices of its information systems, it is important that the IS auditor understand how an organization evaluates, develops, implements, maintains and disposes of its information systems and related components.

Note: A CISA candidate should have a sound understanding of the information systems (hardware and software) acquisition, development and implementation process. This understanding should extend beyond a definitional knowledge of terms and concepts and include an ability to identify vulnerabilities and risk and recommend appropriate controls to effectively mitigate risk. A thorough understanding of the phases of project management is also required. In addition, a CISA candidate should have a good understanding of various application systems and architectures, and the related processes, risk and controls.

3.1 PROJECT GOVERNANCE AND MANAGEMENT

In any organization, several projects are typically running concurrently. In order to identify the relationships among those projects, a common approach is to establish a project portfolio and/or a program management structure. This assists in identifying common objectives for the business organization, identifying and managing risk, and identifying resource connections.

All projects require governance structures, policies and procedures, and specific control mechanisms to assure strategic and tactical alignment with

the organization's respective goals, objectives and risk management strategy. Without proper governance, all aspects of a project may be compromised. Project governance structures should involve the project and the functional line organization. All governance decisions about the project should be driven through the business case, and there must be periodic review of the benefits achieved.

Effective and efficient project management requires that projects be managed based on hard factors such as deliverables, quality, costs and deadlines; on soft factors such as team dynamics, conflict resolution, leadership issues, cultural differences and communication; and, finally, on environmental factors such as the political and power issues in the sponsoring organization, managing the expectations of stakeholders, and the larger ethical and social issues that may surround a project.

Project management structures are dependent on the size of the organization and complexity of business/operations. Accordingly, some roles and responsibilities may be grouped or restructured.

Under such circumstances, the role of an IS auditor is to ensure that rules of system development, as they relate to segregation of duties (SoD) and responsibilities, are not compromised.

There are many approaches and standards to project management. Because there are significant differences in scope, content and wording in each of these approaches and standards, IS auditors must be familiar with the standard in use in their organization prior to involvement in specific projects.

Note: CISA candidates will not be tested on their knowledge of any particular project management approach or standard. However, they must understand the basic elements of project management structures, policies and procedures, and, more specifically, related controls.

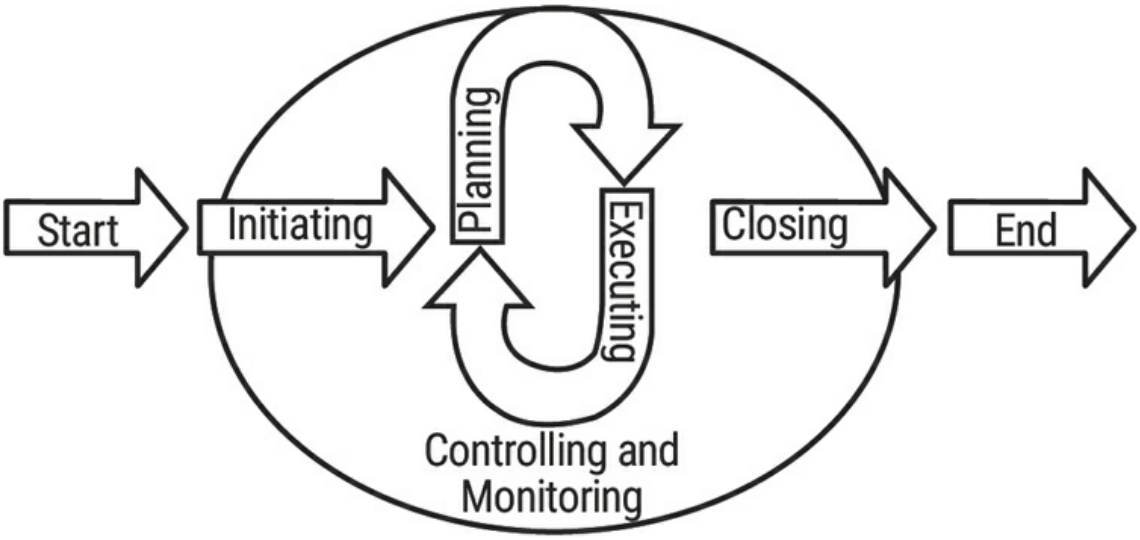
3.1.1 PROJECT MANAGEMENT PRACTICES

Project management is the application of knowledge, skills, tools and

techniques to a broad range of activities to achieve a stated objective such as meeting the defined user requirements, budget and deadlines for an IS project. Project management is a business process in a project-oriented organization. The project management process begins with the project charter and ends with the completion of the project. Project management knowledge and practices are best described in terms of their component processes of initiating, planning, executing, controlling, monitoring and closing a project.

A typical project management process is depicted in **figure 3.1**.

Figure 3.1—Project Management Life Cycle



Source: Adapted from *PMI, A Guide to the Project Management Body of Knowledge 5th Edition*, p. 50

The complexity of project management requires careful and explicit design of the project management process. Thus, all design issues applicable for business process engineering should be applied for the project management process.

3.1.2 PROJECT MANAGEMENT STRUCTURE

Three types of project management organizational structures outline the authority and control within an organization.

1. **Functional-structured organization**—In a functional-structured

organization, the project manager has only a staff function without formal management authority. The work is broken down in departments and a project manager is allowed to advise peers and team members only as to which activities should be completed.

2. **Project-structured organization**—In a project-structured organization, the project manager has formal authority over those taking part in the project. This includes authority over the project's budget, schedule and team.
3. **Matrix-structured project organization**—In a matrix-structured organization, management authority is shared between the project manager and the department heads.

IS projects may be initiated from within any part of the organization, including the IT department. An IS project has specific objectives, deliverables, and start and end dates. Most IS projects are divisible into explicit phases (e.g., SDLC).

Requests for major projects should be submitted to and prioritized by an IT steering committee. The committee then identifies a project manager. The project manager, who need not be an IT staff member, should be given complete operational control over the project and be allocated the appropriate resources, including IS professionals and other staff from user departments, for the successful completion of the project. An IS auditor may be included in the project team as control expert. The IS auditor may also provide an independent, objective review to ensure that the level of involvement (commitment) of the responsible parties is appropriate. In such cases, the IS auditor is not performing an audit but is participating on the project in an advisory role. Depending on the level of the IS auditor's involvement, he/she may become ineligible to perform audits of the application when it becomes operational. An example of a project's organizational structure is shown in **figure 3.2**.

3.1.3 PROJECT MANAGEMENT ROLES AND RESPONSIBILITIES

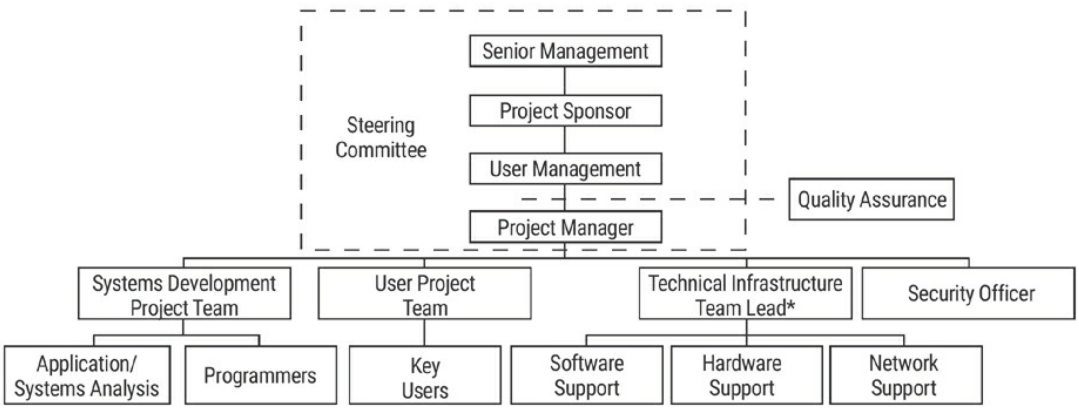
The various roles and responsibilities of groups/individuals that may be

involved in the project management process follow:

- **Project steering committee**—Provides overall direction and ensures appropriate representation of the major stakeholders in the project's outcome. The project steering committee is ultimately responsible for all deliverables, project costs and schedules. This committee should be comprised of a senior representative from each business area that will be significantly impacted by the proposed new system or system modification. The project manager should also be a member of this committee. Each member must have the authority to make decisions related to system designs that will affect their respective departments. Generally, a project sponsor assumes the overall ownership and accountability of the project and chairs the steering committee. The project steering committee performs the following functions:
  - Reviews project progress regularly (e.g., semimonthly or monthly) and holds emergency meetings when required.
  - Serves as coordinator and advisor. Members of the committee should be available to answer questions and make user-related decisions about system and program design.
  - Takes corrective action if necessary due to project progress and issues escalated to the committee.
- **Senior management**—Demonstrates commitment to the project and approves the necessary resources to complete the project. This commitment from senior management helps ensure involvement by those needed to complete the project.
- **Project sponsor**—Provides funding for the project and works closely with the project manager to define the critical success factors (CSFs) and metrics for measuring the success of the project. Data and application ownership are assigned to a project sponsor. A project sponsor is typically the senior manager in charge of the primary business unit the application will support.
- **User management**—Assumes ownership of the project and resulting system; allocates qualified representatives to the team; and actively participates in business process redesign, system requirements definition, test case development, acceptance testing and user training. User management should review and approve system deliverables as they are defined and implemented.

- **User project team**—Completes assigned tasks, communicates effectively with the systems developers by actively involving themselves in the development process as subject matter experts (SMEs), works according to local standards and advises the project manager of expected and actual project plan deviations

Figure 3.2—Project Management, Sample Organization Chart



- **Project manager**—Provides day-to-day management and leadership of the project, ensures that project activities remain in line with the overall direction, ensures appropriate representation of the affected departments, ensures that the project adheres to local standards, ensures that deliverables meet the quality expectations of key stakeholders, resolves interdepartmental conflicts, and monitors and controls costs and the project timetable. The project manager may facilitate the definition of the project scope, manage the budget and control the activities via a project schedule. Where projects are staffed by personnel dedicated to the project, the project manager will have a line responsibility for such personnel.
- **Quality assurance (QA)**—Reviews results and deliverables within each phase and at the end of each phase and confirms compliance with requirements. The objective of this group is to ensure the quality of the project by measuring the adherence of the project staff to the organization's SDLC, advise on deviations, and propose recommendations for process improvements or greater control points when deviations occur. The points where reviews occur depend on the SDLC methodology used, the structure and magnitude of the system and the impact of potential deviations.

Additionally, focus may include a review of appropriate, process-based activities related to either project management or the use of specific software engineering processes within a particular life cycle phase. Such a focus is crucial to completing a project on schedule and within budget and in achieving a given software process maturity level.

- **Systems development management**—Provides technical support for hardware and software environments by developing, installing and operating the requested system. This area also provides assurance that the system is compatible with the organization's computing environment and strategic IT direction and assumes operating support and maintenance activities after installation.
- **Systems development project team**—Completes assigned tasks, communicates effectively with users by actively involving them in the development process, works according to local standards and advises the project manager of necessary project plan deviations
- **Security officer (or security team)**—Ensures that system controls and supporting processes provide an effective level of protection, based on the data classification set in accordance with corporate security policies and procedures; consults throughout the life cycle on appropriate security measures that should be incorporated into the system; reviews security test plans and reports prior to implementation; evaluates security-related documents developed in reporting the system's security effectiveness for accreditation; and periodically monitors the security system's effectiveness during its operational life
- **Information system security engineer**—Applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risk associated with these vulnerabilities. Key to meeting this role is defining the needs, requirements, architectures and designs to construct network, platform and application constructs according to the principles of both defense in breadth and security in depth.

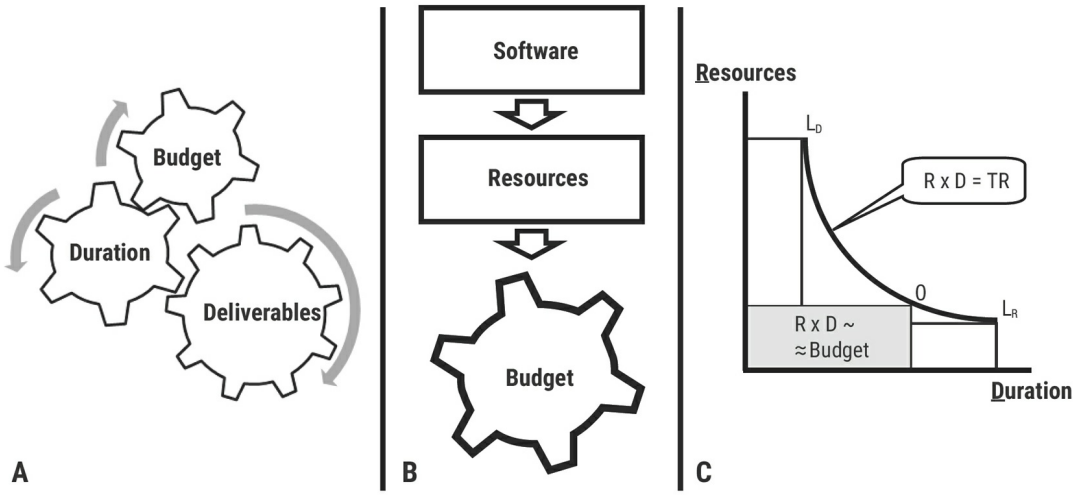
Note: A CISA candidate should be familiar with the general roles and responsibilities of groups or individuals involved in the project management process.

3.1.4 PROJECT MANAGEMENT TECHNIQUES

A number of project management techniques and tools are available to assist the project manager in controlling the time and resources utilized in the development of a system. The techniques and tools may vary from a simple manual effort to a more elaborate computerized process. The size and complexity of the project may require different approaches. Project management techniques also provide systematic quantitative and qualitative approaches to software size estimating, scheduling, allocating resources and measuring productivity. These tools and techniques typically aid in areas described in later sections in this chapter.

There are various elements of a project that should always be considered before selecting a technique. Their relationships are shown in **figure 3.3**.

Figure 3.3—Relationships Between Project Management Elements



Project management should pay attention to three key intertwining elements: deliverables, duration and budget (**figure 3.3A**). Their relationship is very complex but is shown in an oversimplified and schematized manner in the figure. Project duration and budget must be commensurate with the nature and characteristics of the deliverables. In general, there will be a positive correlation between highly demanding deliverables, a long duration and a high budget.

Budget is deduced (**figure 3.3B**) from the resources required to carry out the project by multiplying fees or costs by the amount of each resource. Resources required by the project are estimated at the beginning of the project using techniques of software/project size estimation.

Size estimation yields a “total resources” calculation. Project management decides on the resources allocated at any particular moment in time. In general, it is convenient to assign an (almost) fixed amount of resources, thus probably minimizing costs (direct and administration). **Figure 3.3C** is simplified in its assumption that there is a fixed amount of resources during the entire project. The curve shows resources assigned (R) × duration (D) = total resources (TR, a constant quantity); which is the classic “man × month” dilemma curve. Any point along the curve meets the condition R × D = TR. At any point O on the curve, the area of the rectangle will be TR, proportional to the budget. If few resources are used, the project will take a long time (a point close to L_R); if many resources are used, the project will take a shorter time (a point close to L_D). L_R and L_D are two practical limits: a duration that is too long may not seem possible; use of too many (human) resources at once would be unmanageable.

3.1.5 PORTFOLIO/PROGRAM MANAGEMENT

A project portfolio is defined as all of the projects being carried out in an organization at a given point in time. A program is a group of projects and tasks that are closely linked together through common strategies, objectives, budgets and schedules. Portfolios, programs and projects are often controlled by a project management office (PMO), which governs the processes of project management but are not typically involved in the management of the content. Like projects, programs have a limited time frame (i.e., a defined start and end date) and organizational boundaries. A differentiator is that programs are more complex, usually have a longer duration, a higher budget and higher risk associated with them, and are of higher strategic importance.

A typical IS-related program may be the implementation of a large-scale enterprise resource planning (ERP) system that includes projects that address technology infrastructure, operations, organizational realignment, business

process reengineering (BPR) and optimization, training, and development. Mergers and acquisitions (M&As) may serve as an example of a non–IS-related program that impacts both the gaining and/or divesting organizations' IS architectures and systems, organizational structure, and business processes.

The objective of program management is the successful execution of programs including, but not limited to, the management of program:

- Scope, financials (costs, resources, cash flow, etc.), schedules, objectives and deliverables
- Context and environment
- Communication and culture
- Organization

To make autonomous projects possible while making use of synergies between related projects in the program, a specific program organization is required. Typical program roles are the:

- Program owner
- Program manager
- Program team

The program owner role is distinct from the project owner role. Typical communication structures in a program are program owner's meetings and program team's meetings. Methodology and processes used in program management are very similar to those in project management and run in parallel to each other. However, they must not be combined and have to be handled and carried out separately. To formally start a program, some form of written assignment from the program sponsor (owner) to the program manager and the program team is required. Because programs most often emerge from projects, such an assignment is of paramount importance to set the program context and boundaries as well as formal management authority. In contrast to program management, in which all relevant projects are closely coupled, this is not a requirement in a project portfolio. Projects of a program belong to an organization's project portfolio as do projects that are not associated with a program.

To manage portfolios, programs and projects, an organization requires specific and well-designed structures such as expert pools, a PMO and project portfolio groups. Specific integrative tools such as project management guidelines, standard project plans and project management marketing instruments are also used.

3.1.6 PROJECT MANAGEMENT OFFICE

The PMO, as an owner of the project management and program management process, must be a permanent structure and adequately staffed to provide professional support in these areas to maintain current and develop new procedures and standards. The objective of the PMO is to improve project and program management quality and secure project success, but it can focus only on activities and tasks and not on project or program content.

An IS auditor should be able to differentiate between auditing project content and/or procedural aspects of a program or project. The objectives of project portfolio management are:

- Optimization of the results of the project portfolio (not of the individual projects)
- Prioritizing and scheduling projects
- Resource coordination (internal and external)
- Knowledge transfer throughout the projects

Project Portfolio Database

A project portfolio database is mandatory for project portfolio management. It must include project data such as owner, schedules, objectives, project type, status and cost. Project portfolio management requires specific project portfolio reports. Typical project portfolio reports are a project portfolio bar chart, a profit versus risk matrix and a project portfolio progress graph.

Example:

A. An organization is migrating from legacy applications to an ERP system and has the strategic goal of delivering cutting-edge computers and maintaining high cash flow to continue to fund research and development. To do so, the organization is using its:
   1. Internal pool of application developers to code its strategic business

process of the manufacture of newly designed computers to deliver finished goods and sales order to cash receipts considering the sensitivity of its business model.

2. Vendors to code the nonstrategic business processes of procure to pay and financial accounting.

B. The organization is also pursuing a program for outsourcing transactional processes to a third-party service provider for online sales and a payment portal.

In this context, activities A.1, A.2 and B, individually, are projects. Activities A.1 and A.2 represent a single program, as they are part of the single larger activity of migrating from legacy applications to ERP, and activity B is part of another larger program to outsource noncore manufacturing processes. Activities A.1, A.2 and B (assuming these are the only activities underway in the entity) represent the portfolio for the entity.

3.1.7 PROJECT BENEFITS REALIZATION

The objective of benefits realization is to ensure that IT and the business fulfill their value management responsibilities, particularly that:

- IT-enabled business investments achieve the promised benefits and deliver measurable business value.
- Required capabilities (solutions and services) are delivered:
  - On time, both with respect to schedule and time-sensitive market, industry and regulatory requirements
  - Within budget
- IT services and other IT assets continue to contribute to business value.

See chapter 2, Governance and Management of IT, for more details on benefits realization.

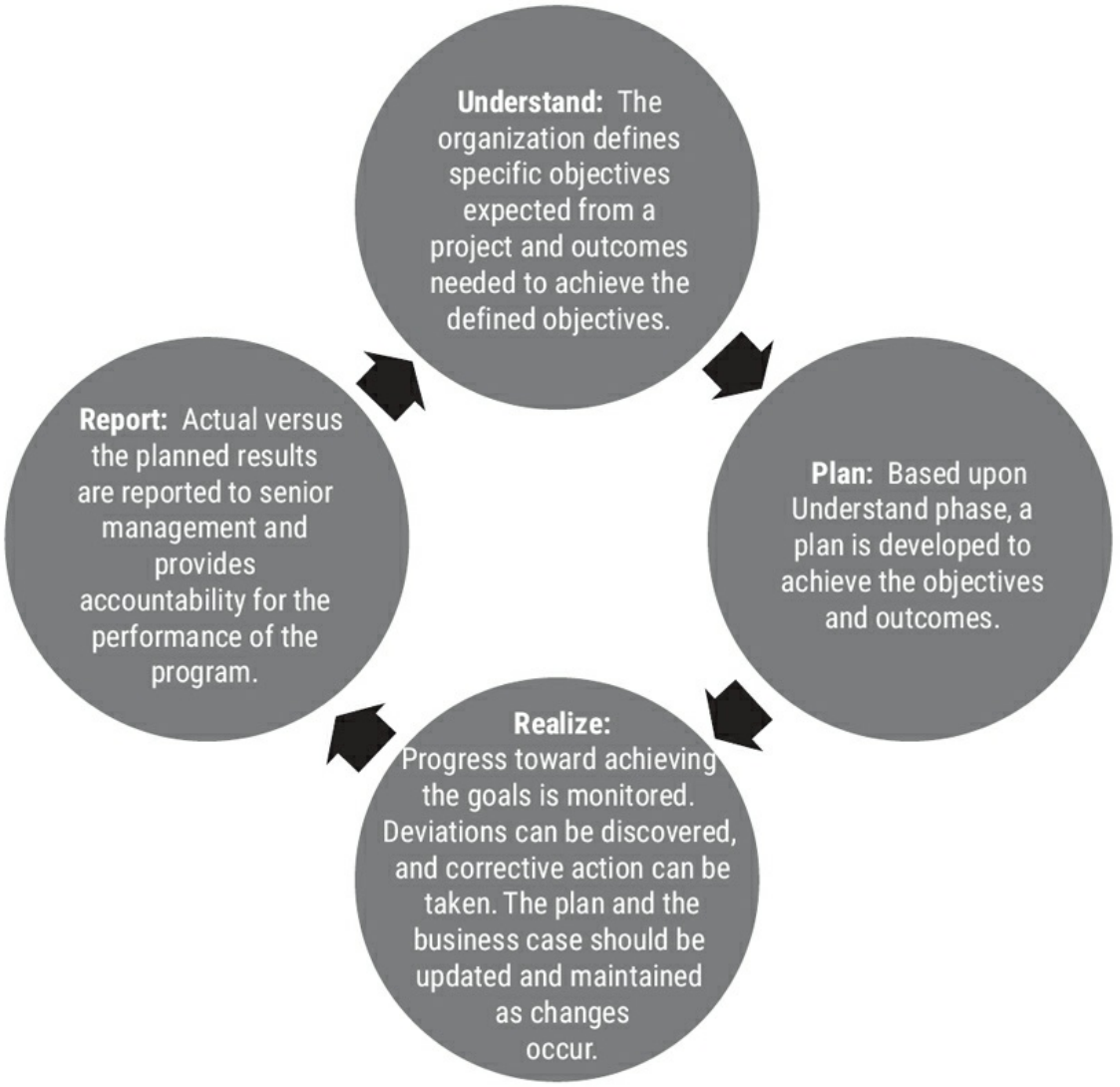
For projects, a planned approach to benefits realization is required, looking beyond project cycles to longer-term cycles that consider the total business benefits and total business costs throughout the life of the new system. Benefits rarely come about exactly as envisioned in plans. An organization has to keep checking and adjusting strategies. Key elements of project

benefits realization are:

- Describing benefits management or benefits realization
- Assigning a measure and target
- Establishing a tracking/measuring regimen
- Documenting the assumption
- Establishing key responsibilities for realization
- Validating the benefits predicted in the business
- Planning the benefit that is to be realized

Generally, benefits realization at the project level encompasses four phases, as indicated in the **figure 3.4**.

Figure 3.4—Benefits Realization Phases—Project Level



Source: New South Wales Government Department of Finance & Services, *Benefits Realisation Guideline, Version 1.2*, Australia, 2011

Project benefits realization is a compromise among major factors such as cost, quality, development/delivery time, reliability and dependability. Strategy-makers perform a comprehensive study and evaluate which factors are “qualifying” or “winning” and then compare those factors with strengths, weaknesses and competencies of services available to complete and maintain systems. Most large organizations employ structured project management principles to support changes to their information systems environment.

As a starting point, an IS auditor should understand how the business defines value or a return on investment (ROI) for development-related projects. If an organizations fails to consistently meet its ROI objectives, this may suggest weakness in its SDLC and related project management practices.

Example:

An organization is planning to invest in an application that would enable customers to manager their orders online. The following would be relevant for the ROI calculation:

A. Business costs
    1. Cost of developing the online application
    2. Cost of controls to ensure integrity of data at rest and in process, while ensuring nonrepudiation
B. Business benefits
    1. Increase in operating profits attributable to expected spike in business driven by customer satisfaction (percent of revenue)
    2. Reduction in operating costs (in terms of dedicated personnel who previously interacted with customers and executed changes)

ROI may be measured as value of benefit/costs, which then can be compared with an organization's cost of funds, to make a go/no-go decision. This ROI framework can then be used as a benchmark to evaluate the progress of the project and identify causes, if the actual ROI is not aligning with the planned ROI.

Project benefits realization is a continuous process that must be managed just like any business process. Assessment of the benefits realization processes and the business case should be a key element of benefits realization processes. Benefits realization often includes a post-implementation review after the implementation of systems. Time must be allowed for initial technical problems to be resolved and for the project benefits to accrue as users become familiar with the new processes and procedures. Project benefits realization must be part of the governance and management of a project and include business sponsorship.

3.1.8 PROJECT INITIATION

A project is initiated by a project manager or sponsor gathering the information required to gain approval for the project to be created. This is often compiled into terms of reference or a project charter that states the objective of the project, the stakeholders in the system to be produced, and the project manager and sponsor. Approval of a project initiation document (PID) or a project request document (PRD) is the authorization for a project to begin. Depending on the size and complexity of the project and the affected parties, the initiation of a project may be achieved by:

- **One-on-one meetings**—One-on-one meetings and a project start workshop help to facilitate two-way communication between the project team members and the project manager.
- **Kick-off meetings**—A kick-off meeting may be used by a project manager to inform the team of what has to be done for the project. Communications involving significant project events should be documented as part of the project artifacts (i.e., project charter meeting, kick-off meeting, gate reviews, stakeholder meetings, etc.).
- **Project start workshops**—A preferred method to ensure that communication is open and clear among the project team members is to use a project start workshop to obtain cooperation from all team members and buy-in from stakeholders. This helps develop a common overview of the project and communicates the project culture early in the project.
- **A combination of the three**—An organization may choose to use two or more of these methods to initiate a project.

3.1.9 PROJECT OBJECTIVES

Project objectives are the specific action statements that support attainment of project goals. All project goals will have one or more objectives identified as the actions needed to reach that goal. The project objective is the means to meet the project goal.

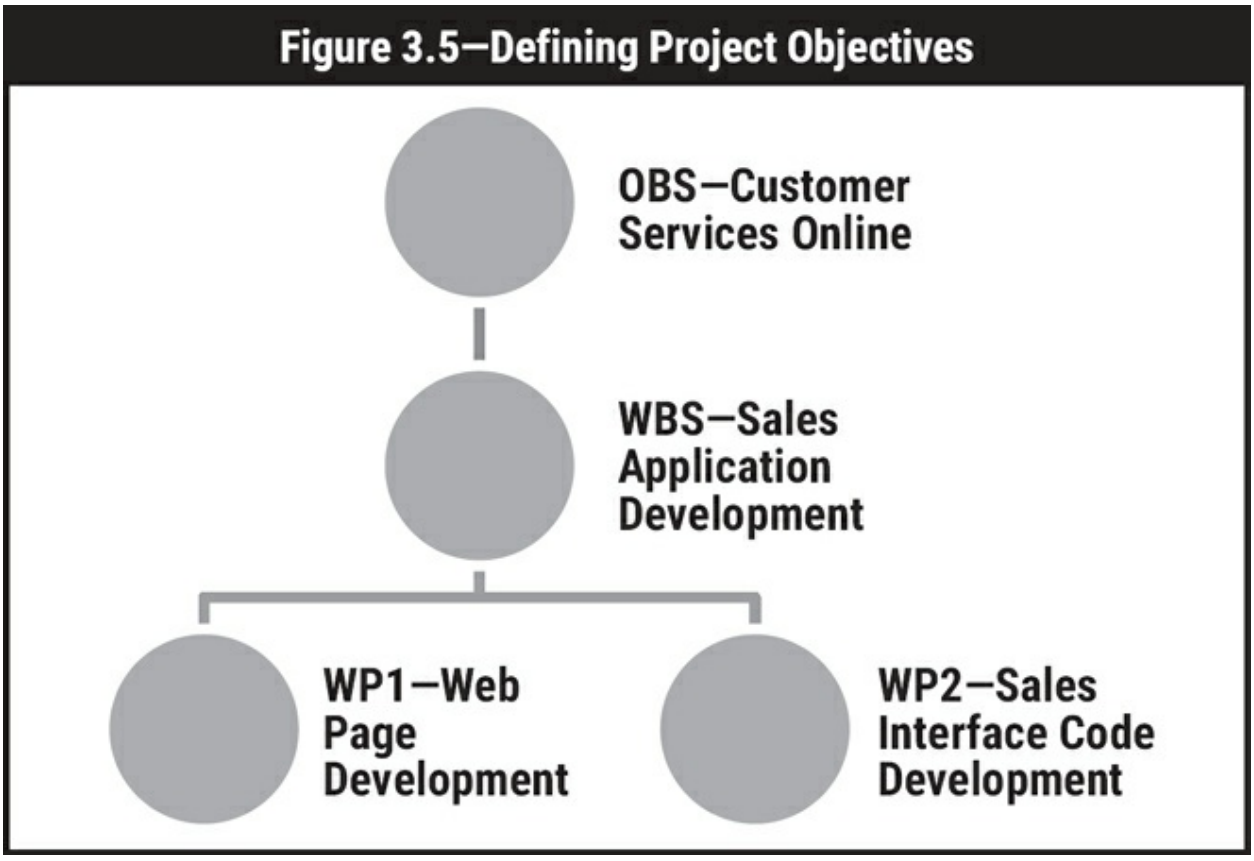
A project must have clearly defined results that are specific, measurable, attainable, realistic and timely (SMART). A commonly accepted approach to define project objectives is to start off with an object breakdown structure (OBS). It represents the individual components of the solution and their

relationships to each other in a hierarchical manner, either graphically or in a table. An OBS can help, especially when dealing with intangible project results such as organizational development, to ensure that a material deliverable is not overlooked.

After the OBS has been compiled or a solution is defined, a work breakdown structure (WBS) is designed to structure all the tasks that are necessary to build up the elements of the OBS during the project. The WBS represents the project in terms of manageable and controllable units of work, serves as a central communications tool in the project, and forms the baseline for cost and resource planning.

In contrast to the OBS, the WBS does not include basic elements of the solution to build but shows individual work packages (WPs) instead. The structuring of the WBS is process-oriented and in phases. The level of detail of the WBS serves as the basis for the negotiations of detailed objectives among the project sponsor, project manager and project team members.

Figure 3.5 shows an example of this process.



Detailed specifications regarding the WBS can be found in WPs. Each WP must have a distinct owner and a list of main objectives and may have a list of additional objectives and out-of-scope objectives. The WP specifications should include dependencies on other WPs and a definition of how to evaluate performance and goal achievement. An example of a WBS is shown in **figure 3.6**.

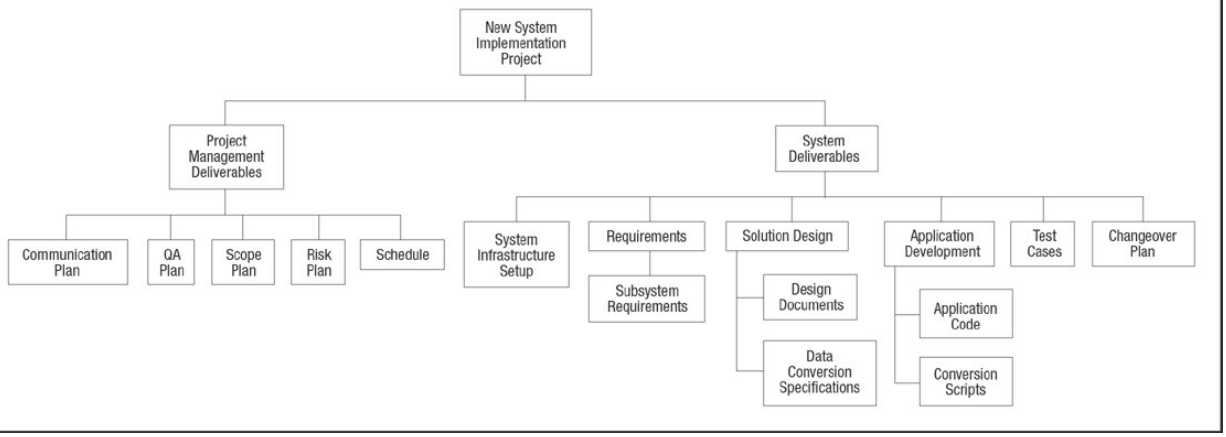
Key things to remember with WBS and respective WPs include the following:

- The top WBS level represents the final deliverable or project.
- Subdeliverables contain WPs that are assigned to an organization's department or unit.
- All elements of the WBS do not need to be defined to the same level.
- WPs define the work, duration and costs for the tasks required to produce the subdeliverable.
- WPs should not exceed a duration of 10 days.
- WPs need to be independent of each other in the WBS.

- WPs are unique and should not be duplicated across the WBS.

To support communications, task lists are often used. A task list is a list of actions to be carried out in relation to WPs and includes assigned responsibilities and deadlines. The task list aids the individual project team members in operational planning and in making agreements. These task lists are most typically compiled into a project schedule at the planning phase of a project and are used in the controlling phase of the project to monitor and track the progress and completion of the WPs. Project schedules are living documents and should indicate the tasks for a WP, the start and finish dates, percentage completed, task dependencies and resource names of individuals planned to work on those tasks. A project schedule will also indicate the stage boundaries explained in section 3.2, Business Case and Feasibility Analysis.

Figure 3.6—Sample Work Breakdown Structure



3.1.10 PROJECT PLANNING

A project must be planned and controlled. A project manager should determine the following:

- Scope of the project (with agreement from stakeholders on project scope)
- Various tasks that need to be performed to produce the expected business application system
- Sequence or order in which these tasks need to be performed
- Duration or the time window for each task
- Priority of each task

- IT and non-IT supporting resources that are available and required to perform these tasks
- Budget or costing for each of these tasks
- Source and means of funding for labor, services, materials, and plant and equipment resources involved in the project

Several different sizing and measurement techniques are used and described in the sections that follow.

Information System Development Project Cost Estimation

Normally relatively large in scope and size, an information system development project focuses on a more complete and integrated solution (hardware, software, facilities, services, etc.) Therefore, these types of projects require much greater planning with regard to estimating and budgeting.

Four commonly used methodologies to estimate the cost of an information system acquisition and development project are:

- **Analogous estimating**—By using estimates from prior projects, the project manager can develop the estimated cost for a new project. This is the quickest estimation technique.
- **Parametric estimating**—The project manager looks at the same past data that were used in analogous estimating and leverages statistical data (estimated employee hours, materials costs, technology, etc.) to develop the estimate. This approach is more accurate than analogous estimation.
- **Bottom-up estimating**—In this method, the cost of each activity in the project is estimated to the greatest detail (i.e., starting at the bottom), and then all the costs are added to arrive at the cost estimate of the entire project. While the most accurate estimate, this is the most time-consuming approach.
- **Actual costs**—Like analogous estimation, this approach takes an extrapolation from the actual costs that were incurred on the same system during past projects.

Software Size Estimation

Software size estimation relates to methods of determining the relative

physical size of the application software to be developed. Estimates can be used to guide the allocation of resources, judge the time and cost required for its development, and compare the total effort required by the resources.

Traditionally, software sizing has been performed using single-point estimations (based on a single parameter) such as source lines of code (SLOC). For complex systems, single-point estimation techniques have not worked because they do not support more than one parameter in different types of programs, which, in turn, affects the cost, schedule and quality metrics. To overcome this limitation, multiple-point estimations have been designed.

Current technologies now take the form of more abstract representations such as diagrams, objects, spreadsheet cells, database queries and graphical user interface (GUI) widgets. These technologies are more closely related to "functionality" deliverables rather than "work" or lines that need to be created.

Function Point Analysis

The function point analysis (FPA) technique is a multiple-point technique used for estimating complexity in developing large business applications.

The results of FPA are a measure of the size of an information system based on the number and complexity of the inputs, outputs, files, interfaces and queries with which a user sees and interacts. This is an indirect measure of software size and the process by which it is developed versus direct size-oriented measures such as SLOC counts.

Function points (FPs) are computed by first completing a table (**figure 3.7**) to determine whether a particular entry is simple, average or complex. Five FP count values are defined, including the number of user inputs, user outputs, user inquiries, files and external interfaces.

Upon completion of the table entries, the count total in deriving the function point is computed through an algorithm that considers complexity adjustment values (i.e., rating factors) based on responses to questions related to issues

such as reliability, criticality, complexity, reusability, changeability and portability. Function points derived from this equation are then used in a manner analogous to SLOC counts as a measure for cost, schedule, productivity and quality metrics (e.g., productivity = FP/person-month, quality = defects/FP, and cost = $/FP).

FPA is an indirect measurement of the software size.

> **Note:** The CISA candidate should be familiar with the use of FPA; however, the CISA exam does not test the specifics on how to perform an FPA calculation.

FPA behaves reasonably well in estimating business applications but not as well for other types of software (such as OS, process control, communications and engineering). Other estimation methods are more appropriate for such software and include the constructive cost model (COCOMO) and FPA feature points of De Marco and Watson-Felix.

Cost Budgets

A system development project should be analyzed with a view to estimating the amount of effort that will be required to carry out each task. The estimates for each task should contain some or all of the following elements:

- Personnel hours by type (e.g., system analyst, programmer, clerical)
- Machine hours (predominantly computer time as well as duplication facilities, office equipment and communication equipment)
- Other external costs such as third-party software, licensing of tools for the project, consultant or contractor fees, training costs, certification costs (if required), and occupation costs (if extra space is required for the project)

Having established a best estimate of expected work efforts by task (i.e., actual hours, minimum/maximum) for personnel, costs budgeting now becomes a two-step process to achieve the following results:

1. Obtain a phase-by-phase estimate of human and machine effort by summing the expected effort for the tasks within each phase.
2. Multiply the effort expressed in hours by the appropriate hourly rate to

obtain a phase-by-phase estimate of systems development expenditure.

Other costs may require tenders or quotes.

Software Cost Estimation

Cost estimation is a result of software size estimation and helps to properly scope a project. Automated techniques for cost estimation of projects at each phase of information system development are available. To use these products, an information system is usually divided into main components and a set of cost drivers is established. Components include:

- Source code language
- Execution time constraints
- Main storage constraints
- Data storage constraints
- Computer access
- The target machine used for development
- The security environment
- Staff experience

After all the drivers are defined, the program will develop cost estimates of the information system and total project.

Scheduling and Establishing the Time Frame

While budgeting involves totaling the human and machine effort involved in each task, scheduling involves establishing the sequential relationship among tasks. This is achieved by arranging tasks according to the following two elements:

Figure 3.7—Computing Function Point Metrics

Measurement Parameter	Count	Weighting Factor			
		Simple	Average	Complex	Results
Number of user inputs		× 3	4	6	=_____
Number of user outputs		× 4	5	7	=_____
Number of user inquiries		× 3	4	6	=_____
Number of files		× 7	10	15	=_____
Number of external interfaces		× 5	7	10	=_____
Count total:					

Note: Organizations that use FP methods develop criteria for determining whether a particular entry is simple, average or complex.

- Earliest start date, by considering the logical sequential relationship among tasks and attempting to perform tasks in parallel, wherever possible
- Latest expected finish date, by considering the estimate of hours per the budget and the expected availability of personnel or other resources, and allowing for known, elapsed-time considerations (e.g., holidays, recruitment time, full-time/part-time employees)

The schedule can be graphically represented using various techniques such as Gantt charts, the critical path method (CPM) or program evaluation and review technique (PERT) diagrams. During the project execution, the budget and schedule should be revisited to verify compliance and identify variances at key points and milestones. Any variances to the budget and schedule should be analyzed to determine the cause and corrective action to take in minimizing or eliminating the total project variance. Variances and the variance analysis should be reported to management on a timely basis.

Gantt Charts

Gantt charts (**figure 3.8**) aid in scheduling the activities (tasks) needed to complete a project. The charts show when an activity should begin and when it should end along a timeline. The charts also show which activities can occur concurrently and which must be completed sequentially. Gantt charts can also reflect the resources assigned to each task and by what percent allocation, as well as aiding in identifying activities that have been completed early or late by comparison to a baseline. Progress of the entire project can be ascertained to determine whether the project is behind, ahead or on schedule compared to the baseline project plan. Gantt charts can also be used to track the achievement of milestones or significant accomplishments for the project such as the end of a project phase or completion of a key deliverable.

Critical Path Methodology

The critical path is the sequence of activities that produces the longest path through a project. All project schedules have (at least) one critical path, usually only one in nonmanipulated project schedules. Critical paths are important because, if everything goes according to schedule, they help estimate the shortest possible completion time for the overall project.

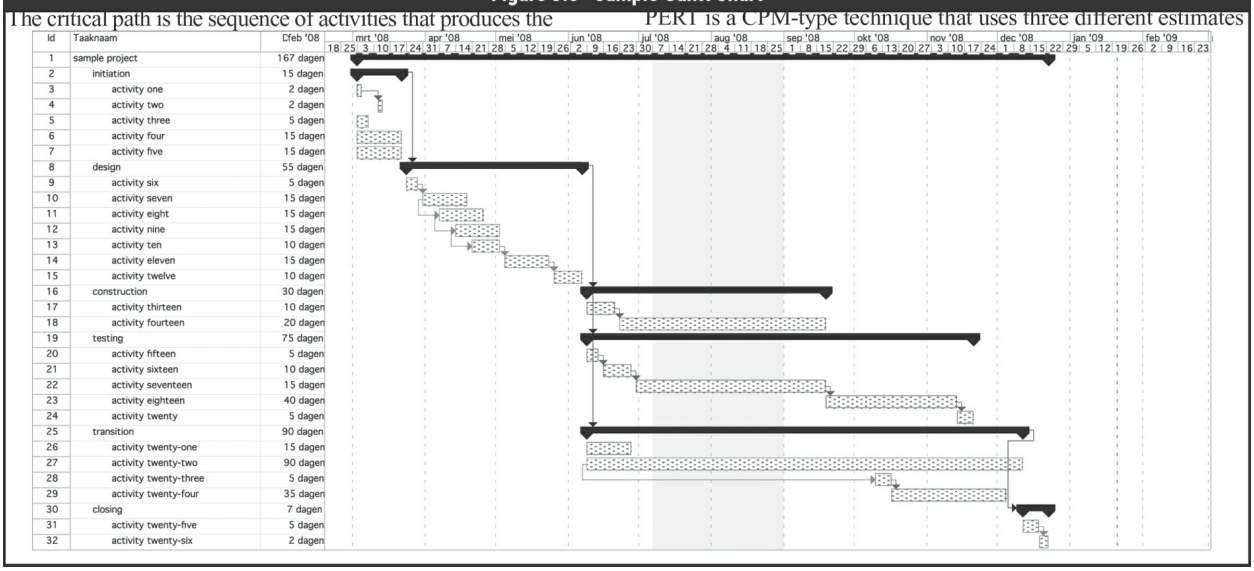
Activities that are not in the critical path have slack time, which is the difference between the latest possible completion time of each activity that will not delay the completion of the overall project and the earliest possible completion time based on all predecessor activities. Activities on a critical path have zero slack time. By working through the network forwards and backwards, the earliest possible completion time for the activities and the project are determined.

All project schedules have a critical path. Because the activities of a project are ordered and independent, a project can be represented as a network in which activities are shown as branches connected at nodes immediately preceding and immediately following activities. A path through the network is any set of successive activities that go from the beginning to the end of the project. Associated with each activity in the network is a single number that best estimates the amount of time the activity will consume.

Most CPM packages facilitate the analysis of resource utilization per time unit (e.g., day, week) and resource leveling, which is a way to level off resource peaks and valleys. Resource peaks and valleys are expensive due to management, hiring, firing, and/or overtime and idle resource costs. A constant, base resource utilization is preferable. There are few, if any, scientific (algorithmic) resource-leveling methods available, but there is a battery (which CPM packages offer) of efficient heuristic methods that yield satisfactory results.

Figure 3.8—Sample Gantt Chart

The critical path is the sequence of activities that produces the PERT is a CPM-type technique that uses three different estimates



Id	Taaknaam	Dfeb '08
1	sample project	167 dagen
2	initiation	15 dagen
3	activity one	2 dagen
4	activity two	2 dagen
5	activity three	5 dagen
6	activity four	15 dagen
7	activity five	15 dagen
8	design	55 dagen
9	activity six	5 dagen
10	activity seven	15 dagen
11	activity eight	15 dagen
12	activity nine	15 dagen
13	activity ten	10 dagen
14	activity eleven	15 dagen
15	activity twelve	10 dagen
16	construction	30 dagen
17	activity thirteen	10 dagen
18	activity fourteen	20 dagen
19	testing	75 dagen
20	activity fifteen	5 dagen
21	activity sixteen	10 dagen
22	activity seventeen	15 dagen
23	activity eighteen	40 dagen
24	activity twenty	5 dagen
25	transition	90 dagen
26	activity twenty-one	15 dagen
27	activity twenty-two	90 dagen
28	activity twenty-three	5 dagen
29	activity twenty-four	35 dagen
30	closing	7 dagen
31	activity twenty-five	5 dagen
32	activity twenty-six	2 dagen

Program Evaluation Review Technique

PERT is a CPM-type technique that uses three different estimates of each activity duration. The three estimates are reduced to a single number (by applying a mathematical formula), and then the classic CPM algorithm is applied. PERT is often used in system development projects with uncertainty about the duration (e.g., pharmaceutical research or complex software development). A diagram illustrating the use of the PERT network management technique is shown in **figure 3.9**, in which events are points in time or milestones for starting and completing activities (arrows). To determine a task's completion, three estimates are shown for completing each activity. The first is the best-case (optimistic) scenario and the third is the worst-case (pessimistic) scenario. The second is the most likely scenario. This estimate is based on experience attained from projects similar in size and scope. To calculate the PERT time estimate for each given activity, the following calculation is applied:

$$[\text{Optimistic} + \text{Pessimistic} + 4(\text{most likely})]/6$$

Using PERT, a critical path is also derived. The critical path is the longest path through the network (only one critical path in a network). The critical path is the route along which the project can be shortened (accelerated) or lengthened (delayed). In **figure 3.9**, the critical path is A, C, E, F, H and I.

The advantage of PERT over CPM in the example provided is that the formula is based on the reasonable assumption that the three time estimates follow a beta statistical distribution and, accordingly, probabilities (with associated confidence levels) can be associated with the total project duration.

When designing a PERT network for system development projects, the first step is to identify all the activities and related events/milestones of the project and their relative sequence. For example, an event or result may be the completion of the operational feasibility study or the point at which the user accepts the detailed design. The analyst must be careful not to overlook any activity. Additionally, some activities such as analysis and design must be preceded by others before program coding can begin. The list of activities determines the detail of the PERT network. The analyst may prepare many diagrams that provide increasingly detailed time estimates.

Timebox Management

Timebox management is a project management technique for defining and deploying software deliverables within a relatively short and fixed period of time and with predetermined specific resources. There is a need to balance software quality and meet the delivery requirements within the timebox. Timebox management can be used to accomplish prototyping or rapid application development-type approaches in which key features are to be delivered in a short time frame. Key features include interfaces for future integrations. The major advantage of this approach is that it prevents project cost overruns and delays from scheduled delivery. The project does not necessarily eliminate the need for a quality process. The design and development phase is shortened due to the use of newer developmental tools and techniques. Preparation of test cases and testing requirements are easily written down as a result of end-user participation. System test and user acceptance testing (UAT) are normally performed together.

3.1.11 PROJECT EXECUTION

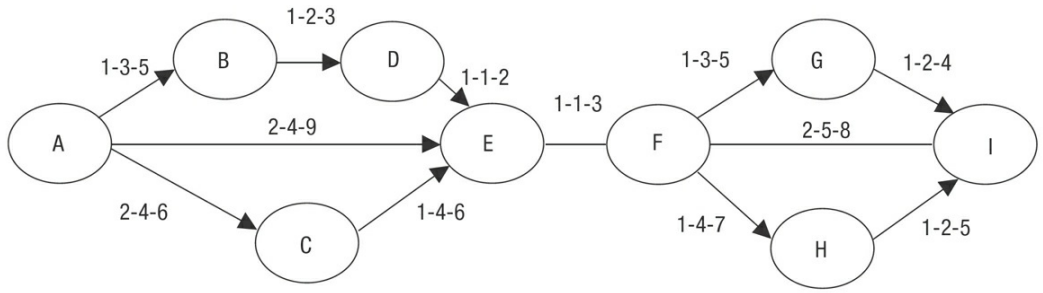
Once planning efforts have been completed, the program manager, in coordination with the PMO, starts the actual project execution of the planned tasks as described in the plans, processes and procedures. The program and

project management team initiates monitoring of internal team production and quality metrics and monitors these metrics from contractors and vendors. A key success factor is the project's oversight of the integrated team in the IT system requirements, architecture, design, development, testing, implementation and transitioning to production operations.

3.1.12 PROJECT CONTROLLING AND MONITORING

The controlling and monitoring activities of a project include management of scope, resource usage and risk. It is important that new requirements for the project be documented and, if approved, allocated appropriate resources. Control of change during a project ensures that projects are completed within stakeholder requirements of time, use of funds and quality objectives. Stakeholder satisfaction should be addressed with effective and accurate requirements capture, proper documentation, baselining and skilled steering committee activity.

Figure 3.9—PERT Network-based Chart



To monitor and measure the development effort, metrics are required. The first step is to identify resources (e.g., people with requisite skills, development tools, facilities) for information system and software development. This will help in estimating and budgeting system and software development resources.

Management of Scope Changes

Managing the scope of projects requires careful documentation in the form of a WBS. This documentation forms part of the project plan or the project baseline. Changes to the scope almost invariably lead to changes in required

activities and impact deadlines and budget. Therefore, it is necessary to have a change management process, including a formal change request submitted to the project manager. Only stakeholders are allowed to submit change requests. Copies of all change requests should be archived in the project file. The project manager judges the impact of each change request on project activities (scope), schedule and budget. The change advisory board then evaluates the change request (on behalf of the sponsor) and decides whether to recommend the change. If the change is accepted, the project manager is instructed to update the project plan to reflect the requested change. The updated project plan must be formally confirmed by the project sponsor—accepting or rejecting the recommendation of the change advisory board.

Management of Resource Usage

Resource usage is the process by which the project budget is being spent. To determine whether actual spending is in line with planned spending, resource usage must be measured and reported. In addition to spending, productivity must be monitored to determine if resource allocation is on task. Whether this is actually happening can be checked with a technique called earned value analysis (EVA).

EVA consists of comparing the metrics at regular intervals during the project such as:

- Budget to date
- Actual spending to date
- Estimate to complete
- Estimate at completion

For example, if a single-task project is planned to take three working days, with eight hours spent each day, the resource will have spent eight hours after the first day, with 16 hours remaining. In order to know if the project is on track, the employee should be asked how much additional time is required to complete the task. If the answer exceeds 16, the project is overrun and not on track.

Management of Risk

Risk is defined as an uncertain event or condition that would impact relevant

aspects of the project. There are two main categories of project risk: the category that impacts the business benefits (and, therefore, endangers the reasons for the project's very existence) and the category that impacts the project itself. The project sponsor is responsible for mitigating the first category of risk and the project manager is responsible for mitigating the second category.

See chapter 2, Governance and Management of IT, for more information.

3.1.13 PROJECT CLOSING

A project has a finite life so, at some point, it must be closed, and a new or modified system is handed over to the users and/or system support staff. At this point, any outstanding issues will need to be assigned. The project sponsor should be satisfied that the system produced is acceptable and ready for delivery.

Key questions to consider include:

- When will the project manager issue the final project closure notification?
- Who will issue the final project notification?
- How will the project manager assist the project team transition to new projects or release them to their regular assigned duties?
- What will the project manager do for actions, risk and issues that remain open? Who will pick up these actions and how will these be funded?

Hand-off of relevant documentation and duties will occur at this stage, and the project team and other relevant stakeholders will identify lessons learned from the project.

Review may be a formal process such as a post-project review. A post-implementation review, in contrast, is typically completed after the project has been in use (or in "production") for some time—long enough to realize its business benefits and costs and measure the project's overall success and impact on the business units. Metrics used to quantify the value of the project include total cost of ownership (TCO) and ROI.

3.1.14 IS AUDITOR'S ROLE IN PROJECT MANAGEMENT

To achieve a successful project outcome, the audit function should play an active role, where appropriate, in the life cycle development of a new system or business application. This will facilitate efforts to ensure that proper controls are designed and implemented in the new system (e.g., continuous concurrent controls for paperless ecommerce systems). An IS auditor needs to understand the system or application being developed in order to identify potential vulnerabilities and points requiring control. If controls are lacking or the process is disorderly, it is an IS auditor's role to advise the project team and senior management of the deficiencies. It may also be necessary to advise those engaged in information systems acquisition and development activities of appropriate controls or processes to implement and follow.

An IS auditor's role may take place during the project or upon completion. Tasks generally include the following:

- Meet with key systems development and user project team members to determine the main components, objectives and user requirements of the system to identify the areas that require controls.
- Discuss the selection of appropriate controls with systems development and user project team members to determine and rank the major risk to and exposures of the system.
- Discuss references to authoritative sources with systems development and user project team members to identify controls to mitigate the risk to and exposures of the system.
- Evaluate available controls and participate in discussions with systems development and user project team members to advise the project team regarding the design of the system and implementation of controls.
- Periodically meet with systems development and user project team members and review the documentation and deliverables to monitor the systems development process to ensure that controls are implemented, user and business requirements are met, and the systems development/acquisition methodology is being followed. Review and evaluate application system audit trails to ensure that documented controls are in place to address all security, edit and processing controls. Tracking information in a change management system includes:
  - History of all work order activity (date of work order, programmer assigned, changes made and date closed)

  - History of logons and logoffs by programmers
  - History of program deletions
  - Adequacy of SoD and QA activities
- Identify and test existing controls to determine the adequacy of production library security to ensure the integrity of the production resources.
- Review and analyze test plans to determine if defined system requirements are being verified.
- Analyze test results and other audit evidence to evaluate the system maintenance process to determine whether control objectives were achieved.
- Review appropriate documentation, discuss with key personnel and use observation to evaluate system maintenance standards and procedures to ensure their adequacy.
- Discuss and examine supporting records to test system maintenance procedures to ensure that they are being applied as described in the standards.
- Participate in post-implementation reviews.

3.2 BUSINESS CASE AND FEASIBILITY ANALYSIS

A business case provides the information required for an organization to decide whether a project should proceed. Depending on the organization and the size of the investment, the development of a business case is either the first step in a project or a precursor to the start of a project. A business case should adequately describe the business reasons or benefits for a project and be of sufficient detail to describe the justification for initiating and continuing a project. It should answer the question, “Why should this project be undertaken and/or continued?” A business case should be a key element of the decision process throughout the life cycle of any project. If at any stage the business case is thought to no longer be valid, the project sponsor or IT steering committee should consider whether the project should proceed. In a well-planned project, there will be decision points, often called “stage gates” or “kill points,” at which a business case is formally reviewed to ensure that it is still valid. If the business case changes during the course of an IT project, the project should be reapproved through the departmental planning and approval process.

After the initial approval has been given to move forward with a project, an analysis begins to clearly define the need and identify alternatives for addressing the need. This analysis is known as the feasibility study. An initial business case would normally derive from a feasibility study undertaken as part of project initiation/planning. This is an early study of a problem to assess if a solution is practical and meets requirements within established budgets and schedule requirements. A feasibility study will normally include the following six elements:

1. **Project scope**—Definition of the business problem and/or opportunity to be addressed. It should be clear, concise and to the point.
2. **Current analysis**—Definition and establishment of an understanding of a system, a software product, etc. Based on this analysis, it may be determined that the current system or software product is working correctly, some minor modifications are needed, or a complete upgrade or replacement is required. At this point in the process, the strengths and weaknesses of the current system or software product are identified.
3. **Requirements**—Definition of project requirements based upon stakeholder needs and constraints. Defining requirements for software differs from defining requirements for systems. The following are examples of needs and constraints used to define requirements:
   - Business, contractual and regulatory processes
   - End-user functional needs
   - Technical and physical attributes defining operational and engineering parameters
4. **Approach**—Definition of a course of action to satisfy the requirements for a recommended system and/or software solution. This step clearly identifies the alternatives that were considered and the rationale as to why the preferred solution was selected. This is the process wherein the use of existing structures and commercial alternatives are considered (e.g., “build versus buy” decisions).
5. **Evaluation**—Examination of the cost-effectiveness of the project based upon the previously completed elements within the feasibility study. The final report addresses the cost-effectiveness of the approach selected. Elements of the final report include:
   - The estimated total cost of the project if the preferred solution is selected, along with the alternates to provide a cost comparison,

including:
  - Estimate of employee hours required to complete
  - Material and facility costs
  - Vendors and third-party contractors' costs
  - Project schedule start and end dates
- A cost and evaluation summary encompassing cost-benefit analysis, ROI, etc.

6. **Review**—Reviews (formal) of the previously completed elements of the feasibility study to both validate the completeness and accuracy of the feasibility study and render a decision to either approve or reject the project or ask for corrections before making a final decision. The review and report are conducted with all key stakeholders. If the feasibility study is approved, all key stakeholders sign the document. Rationale for rejection of the feasibility study should be explained and attached to the document as part of a lesson-learned list for use in future project studies.

3.2.1 IS AUDITOR'S ROLE IN BUSINESS CASE DEVELOPMENT

An IS auditor should understand how a business defines business cases used during feasibility studies and resultant determinations with regard to ROI for information systems development-related projects. If an organization fails to consistently meet its ROI objectives, this may suggest weaknesses in its system development approach and related project management practices.

An IS auditor plays an important role in the review and evaluation of a feasibility study. This is to ensure that the process and decisions were made in an effective and unbiased manner. The following are tasks typically performed by an IS auditor when reviewing a feasibility study:

- Review and evaluate the criticality of the business and information system process requirements.
- Determine if a solution can be achieved with systems already in place. If not, review the evaluation of alternative solutions for reasonableness.
- Determine the reasonableness of the chosen solution based on their strengths and weaknesses.
- Determine whether all cost justifications/benefits are verifiable and reflect

anticipated costs and benefits to be realized.
- Review the documentation produced for reasonableness.

As it applies to the requirements definition within a feasibility study, an IS auditor should perform the following functions:
- Obtain the detailed requirements definition document and verify its accuracy and completeness through interviews with the relevant user departments.
- Identify the key team members on the project team and verify that all affected user groups have/had appropriate representation.
- Verify that project initiation and cost have received proper management approval.
- Review the conceptual design specifications (e.g., transforms, data descriptions) to ensure that they address the needs of the user.
- Review the conceptual design to ensure that control specifications have been defined.
- Determine whether a reasonable number of vendors received a proposal covering the project scope and user requirements.
- Review the UAT specification.
- Determine whether the application is a candidate for the use of an embedded audit routine. If so, request that the routine be incorporated in the conceptual design of the system.

3.3 SYSTEM DEVELOPMENT METHODOLOGIES

A systems development methodology is a structure that an organization uses to plan and control the development of information systems and software and new business applications. In the face of increasing system complexity and the need to implement new systems more quickly to achieve benefits before the business changes, system and software development practitioners have adopted many ways of organizing information system and software projects.

3.3.1 BUSINESS APPLICATION DEVELOPMENT

Business application development is part of a life cycle process with defined phases applicable to deployment, maintenance and retirement. In this process, each phase is an incremental step that lays the foundation for the next phase,

to ensure effective management control in building and operating business application systems.

A business application system developed will fall in one of two major categories:

- **Organization-centric**—The objective of organization-centric applications is to collect, collate, store, archive and share information with business users and various applicable support functions on a need-to-know basis. Thus, sales data are made available to accounts, administration, governmental levy payment departments, etc. Regulatory levy fulfillment (i.e., tax compliance) is also addressed by the presence of organization-centric applications. Organization-centric application projects usually use the SDLC or other, more detailed software engineering approaches for development.
- **End-user-centric**—The objective of an end-user-centric application is to provide different views of data for their performance optimization. This objective includes decision support systems (DSSs) and geographic information systems (GISs). Most of these applications are developed using alternative development approaches.

A business application development project is generally initiated as a result of one or more of the following situations:

- A new opportunity that relates to a new or existing business process
- A problem that relates to an existing business process
- A new opportunity that will enable the organization to take advantage of technology
- A problem with the current technology
- Alignment of business applications with business partners/industry standard systems and respective interfaces

All these situations are tightly coupled with key business drivers, which are defined as the attributes of a business function that drive the behavior and implementation of that business function to achieve the strategic business goals. Thus, all critical business objectives (as a breakdown of the organizational strategy) have to be translated into key business drivers for all parties involved in business operations during a systems development project.

Objectives should be SMART (see section 3.1.9, Project Objectives) so that general requirements will be expressed in a scorecard form, which allows objective evidence to be collected in order to measure the business value of an application and to prioritize requirements. A key benefit of this approach is that all affected parties will have a common and clear understanding of the objectives and how they contribute to business support. Additionally, conflicting key business drivers (e.g., cost versus functionality) and mutually dependent key business drivers can be detected and resolved in early stages of the project.

Business application projects should be initiated using well-defined procedures or activities as part of a defined process to communicate business needs to management. These procedures often require detailed documentation identifying the need or problem, specifying the desired solution and relating the potential benefits to an organization. All internal and external factors affected by the problem and their effect on an organization should be identified.

3.3.2 SDLC MODELS

Several different SDLC models exist, including:

- Traditional waterfall
- V-shaped
- Iterative

A **traditional SDLC (waterfall) model** and variants of the model normally involve a life cycle verification approach that ensures that potential mistakes are corrected early and not solely during final acceptance testing. This life cycle approach is the oldest and most widely commonly used for developing business applications. This approach works best when a project’s requirements are likely to be stable and well defined. It facilitates the determination of a system architecture relatively early in the development effort. The approach is based on a systematic, sequential approach to system and/or software development. The traditional approach is useful in web applications in which prototypes of screens are necessary to aid in the completion of requirements and design.

The primary advantage of the traditional approach is that it provides a template into which methods for the requirements (i.e., definition, design, programming, etc.) can be placed. However, some of the problems encountered with this approach include the following:

- Unanticipated events that result in iterations, creating problems in implementing the approach
- The difficulty of obtaining an explicit set of requirements from the customer/user as the approach requires
- Managing requirements and convincing the user about the undue or unwarranted requirements in the system functionality, which may lead to conflict in the project
- The necessity of customer/user patience, which is required because under this approach a working version of the system's programs will not be available until late in the project's life cycle
- A changing business environment that alters or changes the customer/user requirements before they are delivered

A **verification and validation model**, also called the V-model, also emphasizes the relationship between development phases and testing levels (**figure 3.10**). The most granular testing—the unit test—occurs immediately after programs have been written. Following this model, testing occurs to validate the detailed design. System testing relates to the architectural specification of the system while final UAT references the requirements.

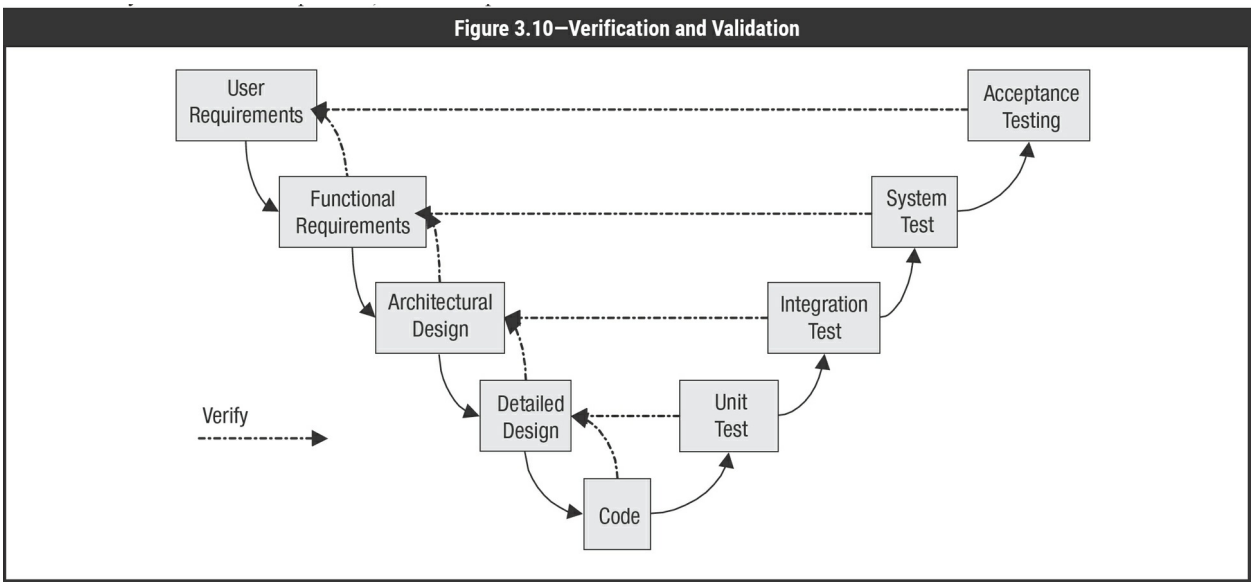
From an IS auditor's perspective, the V-model's defined life cycle phases and specific points for review and evaluation provide the following advantages:

- An IS auditor's influence is significantly increased when there are formal procedures and guidelines identifying each phase in the business application life cycle and the extent of IS auditor involvement.
- An IS auditor can review all relevant areas and phases of the systems development project and report independently to management on the adherence to planned objectives and organizational procedures.
- An IS auditor can identify selected parts of the system and become involved in the technical aspects on the basis of his/her skills and abilities.
- An IS auditor can provide an evaluation of the methods and techniques applied through the development phases of the business application life

cycle.

The **iterative model** is a cyclical process in which business requirements are developed and tested in iterations until the entire application is designed, built and tested. During each iteration, the development process goes through each phase, from requirements through testing, and each subsequent cycle incrementally improves the process.

Figure 3.10—Verification and Validation



Note: A CISA candidate should understand systems development models and how they are used to evaluate the existence and effectiveness of critical system development controls. A CISA candidate should also understand how a selected methodology is used and whether the process will properly address the project requirements.

3.3.3 SDLC PHASES

An SDLC approach is made up of a number of distinct phases (**figure 3.11**), each with a defined set of activities and outcomes. Each phase has defined goals and activities to perform with assigned responsibilities, expected outcomes and target completion dates. There are other interpretations that use a slightly different number of phases with different names.

The actual phases for each project may vary depending on whether a

developed or acquired solution is chosen. For example, system maintenance efforts may not require the same level of detail or number of phases as new applications. The phases and deliverables should be decided during the early planning stages of the project.

Over the years, business application development has occurred largely through the use of traditional SDLC phases. As purchased packages have become more common, the design and development phases of the traditional life cycle are being replaced with selection and configuration phases.

The content that follows describes each phase, its purpose and relationship to prior phases, the general activities performed and expected outcomes.

Phase 1—Feasibility Study

A feasibility study is concerned with analyzing the benefits and solutions for the identified problem area. This study includes development of a business case that states the strategic benefits of implementing the system either in productivity gains or in future cost avoidance, identifies and quantifies the cost savings of the new system, and estimates a payback schedule for the cost incurred in implementing the system or shows the projected ROI.

Figure 3.11—Traditional System Development Life Cycle Approach

SDLC Phase	General Description
Phase 1—Feasibility Study	Determine the strategic benefits of implementing the system either in productivity gains or in future cost avoidance, identify and quantify the cost savings of a new system, and estimate a payback schedule for costs incurred in implementing the system. Further, intangible factors such as readiness of the business users and maturity of the business processes will also be considered and assessed. This business case provides the justification for proceeding to the next phase.
Phase 2—Requirements Definition	Define the problem or need that requires resolution and define the functional and quality requirements of the solution system. This can be either a customized approach or vendor-supplied software package, which would entail following a defined and documented acquisition process. In either case, the user needs to be actively involved.
Phase 3A—Software Selection and Acquisition	Based on requirements defined, prepare a request for proposal outlining the entity requirements to invite bids from prospective

(*purchased systems*)	suppliers, in respect of those systems that are intended to be procured from vendors or solution providers.
Phase 3B—Design (*in-house development*)	Based on the requirements defined, establish a baseline of system and subsystem specifications that describe the parts of the system, how they interface, and how the system will be implemented using the chosen hardware, software and network facilities. Generally, the design also includes program and database specifications and will address any security considerations. Additionally, a formal change control process is established to prevent uncontrolled entry of new requirements into the development process.
Phase 4A—Configuration (*purchased systems*)	Configure the system, if it is a packaged system, to tailor it to the organization's requirements. This is best done through the configuration of system control parameters, rather than changing program code. Modern software packages are extremely flexible, making it possible for one package to suit many organizations simply by switching functionality on or off and setting parameters in tables. There may be a need to build interface programs that will connect the acquired system with existing programs and databases.
Phase 4B—Development (*in-house development*)	Use the design specifications to begin programming and formalizing supporting operational processes of the system. Various levels of testing also occur in this phase to verify and validate what has been developed. This generally includes all unit and system testing and several iterations of user acceptance testing.
Phase 5—Final Testing and Implementation	Establish the actual operation of the new information system, with the final iteration of user acceptance testing and user sign-off conducted in this phase. The system also may go through a certification and accreditation process to assess the effectiveness of the business application in mitigating risk to an appropriate level and providing management accountability over the effectiveness of the system in meeting its intended objectives and in establishing an appropriate level of internal control.
Phase 6—Post-implementation	Following the successful implementation of a new or extensively modified system, implement a formal process that assesses the adequacy of the system and projected cost-benefit or ROI measurements vis-à-vis the feasibility stage findings and deviations. In so doing, IS project and end-user management can provide lessons learned and/or plans for addressing system deficiencies as well as recommendations for future projects regarding system development and project management processes followed.

Intangible benefits such as improved morale may also be identified; however, benefits should be quantified whenever possible.

A feasibility study achieves the following:

- Defines a time frame for the implementation of the required solution
- Determines an optimum alternative risk-based solution for meeting business needs and general information resource requirements (e.g., whether to develop or acquire a system). Such processes can easily be mapped to SDLC and rapid application development (RAD).
- Determines whether an existing system can correct the situation with slight or no modification (e.g., workaround)
- Determines whether a vendor product offers a solution to the problem
- Determines the approximate cost to develop the system to correct the situation
- Determines whether the solution fits the business strategy

Factors impacting whether to develop or acquire a system include the following:

- The date the system needs to be functional
- The cost to develop the system as opposed to buying it
- The resources, staff (availability and skill sets) and hardware required to develop the system or implement a vendor solution
- In a vendor system, the license characteristics (e.g., yearly renewal, perpetual) and maintenance costs
- The other systems needing to supply information to or use information from the vendor system that will need the ability to interface with the system
- Compatibility with strategic business plans
- Compatibility with risk appetite and regulatory compliance needs
- Compatibility with the organization's IT infrastructure
- Likely future requirements for changes to functionality offered by the system

The result of the completed feasibility study should be a comparative report that shows the results of criteria analyzed (e.g., costs, benefits, risk, resources required and organizational impact) and recommends one of the alternatives/solutions and a course of action.

Closely related to a feasibility study is the development of an impact assessment. An impact assessment is a study of the potential future effects of

a development project on current projects and resources. The resulting document should list the pros and cons of pursuing a specific course of action.

Phase 2—Requirements Definition

Requirements definition is concerned with identifying and specifying the business requirements of the system chosen for development during the feasibility study. Requirements include descriptions of:

- What a system should do
- How users will interact with a system
- Conditions under which the system will operate
- Information criteria the system should meet

This phase also deals with overarching issues that are sometimes called nonfunctional requirements (e.g., access control). Many IT security weaknesses can be corrected with a more critical focus on security within the context of the SDLC and, in particular, during the requirements definition.

To successfully complete the requirements definition phase, the project team should perform the following activities:

- Identify and consult stakeholders to determine their requirements.
- Analyze requirements to detect and correct conflicts (mainly, differences between requirements and expectations) and determine priorities.
- Identify system bounds and how the system should interact with its environment.
- Identify any relevant security requirements.
- Convert user requirements into system requirements (e.g., an interactive user interface prototype that demonstrates the screen look and feel).
- Record requirements in a structured format. Historically, requirements have been recorded in a written requirements specification, possibly supplemented by some schematic models. Commercial requirements management tools now are available that allow requirements and related information to be stored in a multiuser database.
- Verify that requirements are complete, consistent, unambiguous, verifiable, modifiable, testable and traceable. Because of the high cost of rectifying requirements' problems in downstream development phases, effective

requirements reviews have a large payoff.
- Resolve conflicts between stakeholders.
- Resolve conflicts between the requirements set and the resources that are available.

The users in this process specify their information resource needs and how they wish to have them addressed by the system (e.g., access controls, regulatory requirements, management information needs and interface requirements).

From this interactive process, a general preliminary design of the system may be developed and presented to user management for review and approval. A project schedule is created for developing, testing and implementing the system. Also, commitments are obtained from the system's developers and affected user departments to contribute the necessary resources to complete the project. It is important to note that all management and user groups must be actively involved in the requirements definition phase to prevent problems such as expending resources on a system that will not satisfy the business requirements. User involvement is necessary to obtain commitment and full benefit from the system. Without management sponsorship, clearly defined requirements and user involvement, the benefits may never be realized.

An IS auditor should pay close attention to the degree the organization's system security engineering team is involved in the development of security controls throughout the data life cycle within the business application. This means the controls are in place regarding applicable confidentiality, integrity and availability of data from creation/receipt to processing, storage, transmission and, ultimately, destruction. This includes whether adequate audit trails are defined as part of the system because these affect the auditor's ability to identify issues for proper follow-up. The IS auditor may also identify regulatory, statutory and legal requirements for the solution being developed.

Phase 3A—Software Selection and Acquisition

At this point in the project, it may be appropriate to evaluate the risk and benefits of developing a new system versus acquiring from a vendor a

suitable system that is complete, tested and proven. Consideration should be given to the ability of the organization to undertake the proposed development project, the costs and risk of doing so, and the benefits of having total ownership and control over the new system rather than becoming dependent on a vendor. Software acquisition is not a phase in the standard SDLC. However, if a decision is reached to acquire rather than develop software, software acquisition is the process that should occur after the requirements definition phase. The decision is generally based on various factors such as the cost differential between development and acquisition, availability of generic software, and the time gap between development and acquisition. Please note that if the result of the decision to develop/acquire is to purchase a vendor-supplied software package, the user must be actively involved in the package evaluation and selection process.

Phase 3B—Design

Based on the general preliminary design and user requirements defined in the requirements definition phase, a detailed design should be developed. Generally, a programming and analyst team is assigned the tasks of defining the software architecture (depicting a general blueprint of the system) and then detailing or decomposing the system into its constituent parts such as modules and components. This approach is an enabler for effective allocation of resources to design the system and define how the system will satisfy all its information requirements. Depending on the complexity of the system, several iterations in defining system-level specifications may be needed to get down to the level of detail necessary to start development activities such as coding.

Key design phase activities include the following:

- Developing system flowcharts and entity relationship models to illustrate how information will flow through the system
- Determining the use of structured design techniques (which are processes to define applications through a series of data or process flow diagrams) that show various relationships from the top level down to the details
- Describing inputs and outputs such as screen designs and reports. If a prototyping tool is going to be used, it is most often used in the screen design and presentation process (via online programming facilities) as part

of an integrated development environment.
- Determining processing steps and computation rules when addressing functional requirement needs
- Determining data file or database system file design
- Preparing program specifications for various types of requirements or information criteria defined
- Developing test plans for the various levels of testing:
  - Unit (program)
  - Subsystem (module)
  - Integration (system)
  - Interface with other systems
  - Loading and initializing files
  - Stress
  - Security
  - Backup and recovery
- Developing data conversion plans to convert data and manual procedures from the old system to the new system. Detailed conversion plans will alleviate implementation problems that arise due to incompatible data, insufficient resources or staff who are unfamiliar with the operations of the new system.
- Performing a risk assessment of information flows

Risk Associated With Software Development

There are many potential types of risk that can occur when designing and developing software systems, including the following:

- **Strategic risk** arises when the business goals are identified and weighted without taking the corporate strategy into account. The strategic importance of the business goals depends on the strategic importance of the related business area.
- **Business risk** (or benefit risk) relates to the likelihood that the new system may not meet the users' business needs, requirements and expectations. For example, the business requirements that were to be addressed by the new system are still unfulfilled, and the process has been a waste of resources. In such a case, even if the system is implemented, it will most likely be underutilized and not maintained, making it obsolete in a short period of time.

- **Project risk** (or delivery risk) arises if the project activities to design and develop the system exceed the limits of the financial resources set aside for the project and, as a result, the project may be completed late, if ever. Software project risk exists at multiple levels:
  - Within the project (e.g., risk associated with not identifying the right requirements to deal with the business problem or opportunity that the system is meant to address and not managing the project to deliver within time and cost constraints)
  - With suppliers (e.g., risk associated with a failure to clearly communicate requirements and expectations, resulting in suppliers delivering late, at more than the expected cost and/or with deficient quality)
  - Within the organization (e.g., risk associated with stakeholders not providing needed inputs or committing resources to the project, and changing organizational priorities and politics)
  - With the external environment (e.g., risk associated with impacts on the projects caused by the actions and changing preferences of customers, competitors, government/regulators and economic conditions)
  - With the technology chosen (e.g., sudden displacement of technology chosen by a one more cost-efficient; insufficient compatibility in the marketplace, resulting in barriers to potential clients' use of the new system)

The primary cause of these problems is a lack of discipline in managing the software development process or the use of a methodology inappropriate to the system being developed. In such instances, an organization is not providing the infrastructure and support necessary to help projects avoid these problems. In such cases, successful projects, if occurring, are not repeatable, and SDLC activities are not defined and followed adequately (i.e., insufficient maturity). However, with effective management, SDLC management activities can be controlled, measured and improved.

An IS auditor should be aware that merely following an SDLC management approach does not ensure the successful completion of a development project. An IS auditor should also review the management discipline over a project related to the following:

- The project meets cooperative goals and objectives.
- Project planning is performed, including effective estimates of resources, budget and time.
- Scope creep is controlled and there is a software baseline to prevent requirements from being added into the software design or having an uncontrolled development process.
- Management is tracking software design and development activities.
- Senior management support is provided to the software project's design and development efforts.
- Periodic review and risk analysis are performed in each project phase.

Use of Structured Analysis, Design and Development Techniques

The use of structured analysis, design and development techniques is closely associated with the traditional, classic SDLC approach to software development. These techniques provide a framework for identifying and representing the data and process components of an application using various graphic notations at different levels of abstraction, until the abstraction level that enables programmers to code the system is reached. Early on, for example, the following activities occur in defining the requirements for a new system:

- Develop system context diagrams (e.g., high-level business process flow schema).
- Perform hierarchical data flow/control flow decomposition.
- Develop control transformations.
- Develop mini-specifications.
- Develop data dictionaries.
- Define all external events—inputs from external environment.
- Define single transformation data flow diagrams (DFDs) from each external event.

The next level of design provides greater detail for building the system, including developing system flowcharts, inputs/outputs, processing steps and computations, and program and data file or database specifications. It should be noted that representation of functions is developed in a modularized top-down fashion. This enables programmers to systematically develop and test modules in a linear fashion.

An IS auditor should be particularly concerned with whether the processes under a structured approach are well defined, documented and followed when using the traditional SDLC approach to business application development.

Entity Relationship Diagrams

An important tool in the creation of a general preliminary design is the use of entity relationship diagrams (ERDs). An ERD depicts a system's data and how these data interrelate. An ERD can be used as a requirements analysis tool to obtain an understanding of the data a system needs to capture and manage. In this case, the ERD represents a logical data model. An ERD can also be used later in the development cycle as a design tool that helps document the actual database schema that will be implemented. Used in this way, the ERD represents a physical data model.

As the name suggests, the essential components of an ERD are entities and relationships. Entities are groupings of like data elements or instances that may represent actual physical objects or logical constructs. An entity is described by attributes, which are properties or characteristics common to all or some of the instances of the entity. Particular attributes, either singularly or in combination, form the keys of an entity. An entity's primary key uniquely identifies each instance of the entity. Entities are represented on ERDs as rectangular boxes with an identifying name.

Relationships depict how two entities are associated (and, in some cases, how instances of the same entity are associated). The classic way of depicting a relationship is a diamond with connecting lines to each related entity. The name in the diamond describes the nature of the relationship. The relationship may also specify the foreign key attributes that achieve the association among the entities. A foreign key is one or more attributes held in one entity that map to the primary key of a related entity.

Software Baselining

The software design phase represents the optimum point for software baselining to occur. The term software baseline means the cutoff point in the design and is also referred to as design freeze. User requirements are reviewed, item by item, and considered in terms of time and cost. The

changes are undertaken after considering various types of risk, and change does not occur without undergoing formal strict procedures for approval based on a cost-benefit impact analysis. Failure to adequately manage the requirements for a system through baselining can result in a number of types of risk. Foremost among these is scope creep—the process through which requirements change during development.

Software baselining also relates to the point when formal establishment of the software configuration management process occurs. At this point, software work products are established as configuration baselines with version numbers. This includes functional requirements, specifications and test plans. All of these work products are configuration items and are identified and brought under formal change management control. This process will be used throughout the application system's life cycle, where SDLC procedures for analysis, design, development, testing and deployment are enforced on new requirements or changes to existing requirements.

User Involvement in the Design

After business processes have been documented and it is understood how those processes might be executed in a new system, involvement of users during the design phase is limited. Given the technical discussion that usually occurs during a design review, end-user participation in the review of detailed design work products is normally not appropriate. However, developers should be able to explain how the software architecture will satisfy system requirements and outline the rationale for key design decisions. Choices of particular hardware and software configurations may have cost implications of which stakeholders need to be aware and control implications that are of interest to an IS auditor.

After the detailed design has been completed, including user approvals and software baselining, the design is distributed to the system developers for coding.

IS Auditor's Role in Project Design

The IS auditor is primarily focused on whether an adequate system of controls is incorporated into system specifications and test plans, and whether

continuous online auditing functions are built into the system (particularly for ecommerce applications and other types of paperless environments). Additionally, an IS auditor is interested in evaluating the effectiveness of the design process itself (such as in the use of structured design techniques, prototyping and test plans, and software baselining) to establish a formal software change process that effectively freezes the inclusion of any changes to system requirements without a formal review and approval process. The key documents coming out of this phase include system, subsystem, program and database specifications, test plans, and a defined and documented formal software change control process.

An IS auditor should perform the following functions:

- Review the system flowcharts for adherence to the general design. Verify that appropriate approvals were obtained for any changes and all changes were discussed and approved by appropriate user management.
- Review for appropriateness the input, processing and output controls designed into the system.
- Interview the key users of the system to determine their understanding of how the system will operate and assess their level of input into the design of screen formats and output reports.
- Assess the adequacy of audit trails to provide traceability and accountability of system transactions.
- Verify the integrity of key calculations and processes.
- Verify that the system can identify and process erroneous data correctly.
- Review the quality assurance results of the programs developed during this phase.
- Verify that all recommended corrections to programming errors were made and the recommended audit trails or embedded audit modules (EAMs) were coded into the appropriate programs.
- Perform a risk assessment.

Phase 4A—Configuration

System configuration, as it relates to the SDLC, consists of defining, tracking and controlling changes in an acquired system to meet the needs of the business. For ERP systems, the task often involves the modification of configuration tables as well as some development, primarily to ensure that

the ERP system is integrated into the existing IT architecture. System configuration is supported by the change management policies and processes, which define:

- Roles and responsibilities
- Classification and prioritization of all changes based on business risk
- Assessment of impact
- Authorization and approval of all changes by the business process owners and IT
- Tracking and status of changes
- Impact on data integrity (e.g., all changes to data files being made under system and application control rather than by direct user intervention)

Phase 4B—Development

The development phase uses the detailed design developed in phase 3B. Responsibilities in this phase rest primarily with programmers and systems analysts who are building the system. Key activities performed in a test/development environment include:

- Coding and developing program and system-level documents
- Debugging and testing the programs developed
- Developing programs to convert data from the old system for use on the new system
- Creating user procedures to handle transition to the new system
- Training selected users on the new system because their participation will be needed
- Ensuring modifications are documented and applied accurately and completely to vendor-acquired software to ensure that future updated versions of the vendor's code can be applied
- Identifying secure coding and configuration standards

Programming Methods and Techniques

To enhance the quality of programming activities and future maintenance capabilities, program coding standards should be applied. Program coding standards are essential to writing, reading and understanding code, simply and clearly, without having to refer back to design specifications. Elements of program coding standards include methods and techniques for internal (source-code level) documentation, methods for data declaration, and an

approach to statement construction and techniques for input/output (I/O). The programming standards applied are an essential control because they serve as a method of communicating among members of the program team, and between the team and users during system development. Program coding standards minimize system development setbacks resulting from personnel turnover, provide the material needed to use the system effectively, and are required for efficient program maintenance and modifications.

Additionally, traditional structured programming techniques should be applied in developing quality and easily maintained software products. They are a natural progression from the top-down structuring design techniques previously described. Like the design specifications, structured application programs are easier to develop, understand and maintain because they are divided into subsystems, components, modules, programs, subroutines and units. Generally, the greater the extent to which each software item described performs a single, dedicated function (cohesion) and retains independence from other comparable items (coupling), the easier it is to maintain and enhance a system because it is easier to determine where and how to apply a change and reduce the chances of unintended consequences.

Online Programming Facilities (Integrated Development Environment)

To facilitate effective use of structured programming methods and techniques, an online programming facility should be available as part of an integrated development environment (IDE). This allows programmers to code and compile programs interactively with a remote computer or server from a terminal or a client's PC workstation. Through this facility, programmers can enter, modify and delete programming codes as well as compile, store and list programs (source and object) on the development computer. The online facilities can also be used by non-IS staff to update and retrieve data directly from computer files.

Online programming facilities are used on PC workstations. The program library is on a server, such as a mainframe library management system, but the modification/development and testing are performed on the workstation. This approach can lower the development costs, maintain rapid response time, and expand the programming resources and aids available (e.g., editing

tools, programming languages, debugging aids). From the perspective of control, this approach introduces the potential weaknesses of:

- The proliferation of multiple versions of programs
- Reduced program and processing integrity through the increased potential for unauthorized access and updating
- The possibility that valid changes could be overwritten by other changes

In general, an online programming facility allows faster program development and helps to enforce the use of standards and structured programming techniques. Online systems improve the programmer's problem-solving abilities, but online systems create vulnerabilities resulting from unauthorized access. Access control software should be used to help reduce the risk.

Programming Languages

Application programs must first be coded in statements, instructions or a programming language that is easy for a programmer to write and that can be read by the computer. These statements (source code) will then be translated by the language translator/compiler into a binary machine code or machine language (object code) that the computer can execute.

Programming languages commonly used for developing application programs include the following:

- High-level, general-purpose programming languages such as COBOL and the C programming language
- Object-oriented languages for business purposes such as C++, Eiffel and Java
- IDEs such as Visual Studio or JBuilder, which provide coding templates automatically
- Web scripting languages (e.g., Hypertext Markup Language [HTML], Javascript, Cascading Style Sheets [CSS])
- Data-centric programming languages (e.g., R, Scala, IDL, SAS)
- Scripting languages such as shell, Perl, Tcl, Python, JavaScript and VBScript. In web development, scripting languages are used commonly to write common gateway interface (CGI) scripts that are used to extend the functionality of web server application software (e.g., to interface with

search engines, create dynamic web pages and respond to user input).
- Low-level assembler languages designed for a specific processor type that are usually used for embedded applications (e.g., slot machines, vending machines, aerospace devices)
- Fourth-generation, high-level programming languages (4GLs), which consist of a database management system (DBMS), embedded database manager, and a nonprocedural report and screen generation facility. 4GLs provide fast iteration through successive designs. Examples of 4GLs include FOCUS, Natural and dBase.
- Decision support or expert systems languages (e.g., EXPRESS, Lisp and Prolog)

Program Debugging

Many programming bugs are detected during the system development process, after a programmer runs a program in the test environment. The purpose of debugging programs during system development is to ensure that all program abends (unplanned ending of a program due to programming errors) and program coding flaws are detected and corrected before the final program goes into production. A debugging tool is a program that will assist a programmer in debugging, fixing or fine-tuning the program under development. Compilers have some potential to provide feedback to a programmer, but they are not considered debugging tools. Debugging tools fall into three main categories:

- **Logic path monitors**—Report on the sequence of events performed by the program, thus providing the programmer with clues on logic errors
- **Memory dumps**—Provide a picture of the internal memory's content at one point in time. This is often produced at the point where the program fails or is aborted, providing the programmer with clues on inconsistencies in data or parameter values. A variant, called a trace, will do the same at different stages in the program execution to show changes in machine-level structures such as counters and registers.
- **Output analyzers**—Help check results of program execution for accuracy. This is achieved by comparing expected results with the actual results.

Phase 5—Final Testing and Implementation

During the implementation phase, the actual operation of the new information

system is established and tested. Final UAT is conducted in this environment. The system may also go through a certification and accreditation process to assess the business application's effectiveness at mitigating risk to an appropriate level and provide management accountability over the effectiveness of the system in meeting its intended objectives and establishing an appropriate level of internal control.

After a successful full-system testing, the system is ready to migrate to the production environment. The programs have been tested and refined; program procedures and production schedules are in place; all necessary data have been successfully converted and loaded into the new system; and the users have developed procedures and been fully trained in the use of the new system. A date for system migration is determined and production turnover takes place. In the case of large organizations and complex systems, this may involve a project in itself and require a phased approach.

Planning for the implementation should begin well in advance of the actual implementation date, and a formal implementation plan should be constructed in the design phase and revised accordingly as development progresses. Each step in setting up the production environment should be stipulated, including who will be responsible, how the step will be verified and the backout procedure if problems are experienced. If the new system will interface with other systems or is distributed across multiple platforms, some final commissioning tests of the production environment may be desirable to prove end-to-end connectivity. If such tests are run, care will be needed to ensure test transactions do not remain in production databases or files.

In the case of acquired software, the implementation project should be coordinated by user management with the help of IS management, if required. The total process should not be delegated to the vendor, to avoid possible unauthorized changes or introduction of malicious code by the vendor's employees/representatives.

After operations are established, the next step is to perform site acceptance testing, which is a full-system test conducted on the actual operations

environment. UAT supports the process of ensuring that the system is production-ready and satisfies all documented requirements. A security test, such as a penetration test, may also be performed at this stage.

Phase 6—Post-implementation Review

Following the successful implementation of a new or extensively modified system, it is beneficial to verify the system has been properly designed and developed and that proper controls have been built into the system. A post-implementation review should meet the following objectives:

- Assessing the adequacy of the system
  - Does the system meet user requirements and business objectives?
  - Have access controls been adequately defined and implemented?
- Evaluating the projected cost benefits or ROI measurements
- Developing recommendations that address the system's inadequacies and deficiencies
- Developing a plan for implementing the recommendations
- Assessing the development project process
  - Were the chosen methodologies, standards and techniques followed?
  - Were appropriate project management techniques used?

3.3.4 IS AUDITOR'S ROLE IN SDLC PROJECT MANAGEMENT

Throughout the project management process an IS auditor should analyze the associated risk and exposures inherent in each phase of the SDLC and ensure that the appropriate control mechanisms are in place to minimize risk in a cost-effective manner. Caution should be exercised to avoid recommending controls that cost more to administer than the associated risk they are designed to minimize.

When reviewing the SDLC process, an IS auditor should obtain documentation from the various phases and attend project team meetings, offering advice to the project team throughout the system development process. An IS auditor should also assess the project team's ability to produce key deliverables by the promised dates.

Typically, an IS auditor should review the adequacy of the following project

management activities:

- Levels of oversight by project committee/board
- Risk management methods within the project
- Issue management
- Cost management
- Processes for planning and dependency management
- Reporting processes to senior management
- Change control processes
- Stakeholder management involvement
- Sign-off process (at a minimum, signed approvals from systems development and user management responsible for the cost of the project and/or use of the system)

Additionally, adequate and complete documentation of all phases of the SDLC process should be evident. Typical types of documentation include, but should not be limited to, the following:

- Objectives defining what is to be accomplished during that phase
- Key deliverables by phases with project personnel assigned direct responsibilities for these deliverables
- A project schedule with highlighted dates for the completion of key deliverables
- An economic forecast for that phase, defining resources and the cost of the resources required to complete the phase

3.3.5 SOFTWARE DEVELOPMENT METHODS

There are several different methods of designing and developing a software system. The choice of a particular method is driven by considerations such as organizational policy, developer knowledge and preference, and the technology being used. The selection of a software development method is generally independent of the selection of a project organization model. An object-oriented approach could be utilized on a project organized into distinct phases, as in a traditional waterfall model of software development, as could an agile project method in which each short iteration delivers working software.

Prototyping—Evolutionary Development

Prototyping, also known as “heuristic” or “evolutionary development,” is the process of creating a system through controlled trial-and-error procedures to reduce the level of risk in developing the system. That is, it enables the developer and customer to understand and react to risk at each evolutionary level (using prototyping as a risk reduction mechanism). It combines the best features of classic SDLC by maintaining the systematic stepwise approach and incorporates it into an iterative framework that more realistically reflects the real world.

The initial emphasis during the development of the prototype is usually placed on the reports and screens, which are the system aspects most used by the end users. This allows the end user to see a working model of the proposed system within a short time. There are two basic methods or approaches to prototyping:

1. Build the model to create the design (i.e., the mechanism for defining requirements). Then, based on that model, develop the system design with all the performance, quality and maintenance features needed.
2. Gradually build the actual system that will operate in production using a 4GL that has been determined to be appropriate for the system being built.

The problem with the first approach is that there can be considerable pressure to implement an early prototype. Often, users observing a working model cannot understand why the early prototype must be refined further. The fact that the prototype needs to be expanded to handle transaction volumes, client-server network connectivity, and backup and recovery procedures, and provide for security, auditability and control is often not understood.

The second approach typically works with small applications using 4GL tools. However, for larger efforts, it is necessary to develop a design strategy for the system even if a 4GL is used. The use of 4GL techniques alone will cause the same difficulties (e.g., poor quality, poor maintainability and low user acceptance) encountered when developing business applications using conventional approaches.

Another overall disadvantage of prototyping is that it often leads to functions or extras being added to the system that are not included in the initial

requirements document. All major enhancements beyond the initial requirements document should be reviewed to ensure that they meet the strategic needs of the organization and are cost-effective. Otherwise, the final system may be functionally rich but inefficient.

A potential risk with prototyped systems is that the finished system will have poor controls. By focusing mainly on what the user wants and what the user sees, system developers may miss some of the controls, such as backup recovery, security and audit trails, that come out of the traditional system development approach.

Change control often becomes much more complicated with prototyped systems. Changes in designs and requirements happen so quickly that they are seldom documented or approved, and the system can escalate to a point of not being maintainable.

Although an IS auditor should be aware of the risk associated with prototyping, an IS auditor should also be aware that this method of system development can provide the organization with significant time and cost savings.

Rapid Application Development

RAD is a methodology that enables an organization to develop strategically important systems quickly while reducing development costs and maintaining quality. This is achieved by using a series of proven application development techniques within a well-defined methodology. These techniques include the use of:

- Small, well-trained development teams
- Evolutionary prototypes
- Integrated power tools that support modeling, prototyping and component reusability
- A central repository
- Interactive requirements and design workshops
- Rigid limits on development time frames

RAD supports the analysis, design, development and implementation of individual application systems. However, RAD does not support the planning

or analysis required to define the information needs of the enterprise as a whole or of a major business area of the enterprise. RAD provides a means for developing systems faster while reducing cost and increasing quality. This is done by automating large portions of the SDLC, imposing rigid limits on development time frames and reusing existing components. The RAD methodology has four major stages:

1. The **concept definition stage** defines the business functions and data subject areas that the system will support and determines the system scope.
2. The **functional design stage** uses workshops to model the system's data and processes and build a working prototype of critical system components.
3. The **development stage** completes the construction of the physical database and application system, builds the conversion system, and develops user aids and deployment work plans.
4. The **deployment stage** includes final-user testing and training, data conversion and the implementation of the application system.

RAD uses prototyping as its core development tool no matter which underlying technology is used. In contrast, object-oriented software development (OOSD) and data-oriented system development (DOSD) use continuously developing models but have a focus on content solution space (e.g., how to best address the problem to make the code reusable and maintainable) and can be applied using a traditional waterfall approach. It should also be noted that BPR attempts to convert an existing business process rather than make dynamic changes.

Agile Development

Agile development is an alternative method for software development. Assuming that all requirements cannot be articulated upfront, agile approaches, such as the Scrum methodology, propose a more iterative and incremental approach instead of the sequential approach of the SDLC. Scrum aims to move planning and directing tasks from the project manager to the team, leaving the project manager to work on removing the obstacles to the team's ability to achieve its objectives. Other agile methods include Extreme Programming (XP), Crystal, Adaptive Software Development (ASD), Feature

Driven Development (FDD) and Dynamic Systems Development Method (DSDM). These processes are termed “agile” because they are designed to flexibly handle changes to the system being developed or the project that is performing the development.

Agile development processes have a number of common characteristics:

- The use of small, time-boxed subprojects or iterations. In this instance, each iteration forms the basis for planning the next iteration.
- Replanning the project at the end of each iteration (referred to as a “sprint” in Scrum), including reprioritizing requirements, identifying any new requirements and determining within which release delivered functionality should be implemented
- Relatively greater reliance, compared to traditional methods, on tacit knowledge—the knowledge in people’s heads—as opposed to external knowledge that is captured in project documentation
- A heavy influence on mechanisms to effectively disseminate tacit knowledge and promote teamwork. Therefore, teams are kept small in size, comprise both business and technical representatives, and are located physically together. Team meetings to verbally discuss progress and issues occur daily, but with strict time limits.
- At least some of the agile methods stipulate pair-wise programming (two persons code the same part of the system) as a means of sharing knowledge and as a quality check.
- A change in the role of the project manager, from one primarily concerned with planning the project, allocating tasks and monitoring progress to that of a facilitator and advocate. Responsibility for planning and control is delegated to the team members.

Agile development does not ignore the concerns of traditional software development but approaches them from a different perspective. Agile development:

- Only plans for the next iteration in detail, rather than planning subsequent development phases.
- Uses an adaptive approach to requirements and does not emphasize managing a requirements baseline.
- Focuses on quickly proving an architecture by building actual functionality

versus formally defining, early on, software and data architecture in increasingly more detailed models and descriptions.
- Assumes limits to defect testing but attempts to validate functions through a frequent-build test cycle and correct problems in the next subproject before too much time and cost are incurred.
- Does not emphasize defined and repeatable processes, but instead performs and adapts its development based on frequent inspections.

Object-oriented System Development

OOSD is the process of solution specification and modeling in which data and procedures can be grouped into an entity known as an object. An object's data are referred to as its attributes and its functionality is referred to as its methods. This contrasts with the traditional (structured SDLC) approach that considers data separately from the procedures that act on them (e.g., program and database specifications). Proponents of OOSD claim the combination of data and functionality is aligned with how humans conceptualize everyday objects.

OOSD is a programming technique, not a software development methodology. OOSD can be done while following any of the widely diverse set of software methodologies. A particular programming language or use of a particular programming technique does not imply or require use of a particular software development methodology.

Objects usually are created from a general template called a "class." The template contains the characteristics of the class without containing the specific data that need to be inserted into the template to form the object. Classes are the basis for most design work in objects. Classes are either superclasses (i.e., root or parent classes) with a set of basic attributes or methods or subclasses, which inherit the characteristics of the parent class and may add (or remove) functionality as required. In addition to inheritance, classes may interact through sharing data, referred to as aggregate or component grouping, or sharing objects.

Aggregate classes interact through messages, which are requests for services from one class (called a client) to another class (called a server). The ability

of two or more objects to interpret a message differently at execution, depending on the superclass of the calling object, is termed “polymorphism.”

To realize the full benefits of using object-oriented programming, it is necessary to employ object-oriented analysis and design approaches. Dealing with objects should permit analysts, developers and programmers to consider larger logical chunks of a system and clarify the programming process.

The major advantages of OOSD are as follows:
- The ability to manage an unrestricted variety of data types
- Provision of a means to model complex relationships
- The capacity to meet the demands of a changing environment

A significant development in OOSD is the use of Unified Modeling Language (UML). UML is a general-purpose notational language for specifying and visualizing complex software for large object-oriented projects, but it may be used for other purposes. This signals a maturation of the object-oriented development approach. While object-orientation is not yet pervasive, it can accurately be said to have entered the computing mainstream. Applications that use object-oriented technology are:
- Web applications
- Ebusiness applications
- Computer-aided software engineering (CASE) for software development
- Office automation for email and work orders
- Artificial intelligence (AI)
- Computer-assisted manufacturing (CAM) for production and process control

Component-based Development

Component-based development can be regarded as an outgrowth of object-oriented development. Component-based development means assembling applications from cooperating packages of executable software that make their services available through defined interfaces (i.e., enabling pieces of programs called “objects,” to communicate with one another regardless of the programming language in which they were written or what OS they are running). The basic types of components are:

- **In-process client components**—These components must run from within a parent/host container such as a mobile application, a virtual machine appliance or an applet.
- **Stand-alone client components**—Applications that expose services to other software can be used as components. Well-known examples are Microsoft's Excel and Word.
- **Stand-alone server components**—Processes running on servers that provide services in standardized ways can be components by means of web application frameworks, application servers, web services, Lightweight Directory Access Protocol (LDAP) directory services, etc.
- **In-process server components**—These components run on servers within containers. Examples include Microsoft's Transaction Server (MTS) and Oracle's JavaBeans.

Tool developers support one or another of these standards with powerful visual tools now available for designing and testing component-based applications. Additionally, a growing number of commercially available application servers support MTS or Enterprise Java Beans (EJB). There is a growing market for third-party components. A primary benefit of component-based development is the ability to buy proven, tested software from commercial developers. The range of components available has increased. The first components were simple in concept (e.g., buttons and list boxes). Components now provide much more diverse functionality. Databases are now available on the web to search for commercial components.

Components play a significant role in web-based applications. Applets are required to extend static HTML, ActiveX controls or Java. Both technologies are compatible with component development.

Component-based development:
- **Reduces development time**—If an application system can be assembled from prewritten components and only code for unique parts of the system needs to be developed, then this should prove faster than writing the entire system from scratch.
- **Improves quality**—Using prewritten components means a significant percentage of the system code has been tested already.

- **Allows developers to focus more strongly on business functionality**—An outcome of component-based development and its enabling technologies is to further increase abstraction already achieved with high-level languages, databases and user interfaces. Developers are shielded from low-level programming details.
- **Promotes modularity**—By encouraging or forcing impassable interfaces between discrete units of functionality, it encourages modularity.
- **Simplifies reuse**—It avoids the need to be conversant with procedural or class libraries, allowing cross-language combination and allowing reusable code to be distributed in an executable format (i.e., no source is required). (To date, large-scale reuse of business logic has not occurred.)
- **Reduces development cost**—Less effort needs to be expended on design and build. Instead, the cost of software components can be spread across multiple users.
- **Supports multiple development environments**—Components written in one language can interact with components written in other languages or running on other machines.
- **Allows a satisfactory compromise between build and buy options**—Instead of buying a complete solution, which perhaps does not entirely fit requirements, it could be possible to purchase only needed components and incorporate these into a customized system.

To realize these advantages, attention to software integration should be provided early and continuously during the development process. No matter how efficient component-based development is, if system requirements are poorly defined or the system fails to adequately address business needs, the project will not be successful.

Web-Based Application Development

Web-based application development is an important software development approach designed to achieve easier and more effective integration of code modules within and between enterprises. Historically, software written in one language on a particular platform has used a dedicated application programming interface (API). The use of specialized APIs has caused difficulties in integrating software modules across platforms. Technologies such as Common Object Request Broker Architecture (CORBA) and COM

that use remote procedure calls (RPCs) have been developed to allow real-time integration of code across platforms. However, using these RPC approaches for different APIs still remains complex. Web-based application development and associated Extensible Markup Language (XML) technologies are designed to further facilitate and standardize code module and program integration.

The other problem that web-based application development seeks to address is avoiding the need to perform redundant computing tasks with the inherent need for redundant code. One obvious example of this is a change of address notification from a customer. Instead of having to update details separately in multiple databases (e.g., contact management, accounts receivable and credit control), it is preferable for a common update process to update the multiple places required. Web services are intended to make this relatively easy to achieve.

Web application development is different from traditional third- or fourth-generation program developments in many ways—from the languages and programming techniques used, to the methodologies used to control the development work, to the way the users test and approve the development work. The risk of application development remains the same. For example, buffer overflows have been a risk since computer programming was invented (e.g., truncation issues with first-generation computer programs), because they are widely known when they could be exploited by almost anyone, almost anywhere in the world, courtesy of the Internet.

As with traditional program development, a risk-based approach should be taken in the assessment of web application vulnerabilities: Identify the business goals and supporting IT goals related to the development, then identify what can go wrong. Previous experience can be used to identify risk related to inadequate specifications, poor coding techniques, inadequate documentation, inadequate quality control (QC) and QA (including testing inadequacies), lack of proper change control and controls over promotion into production, and so on, and put these in the context of the web application languages, development processes and deliverables (perhaps with the support of best practice material/literature on web applications development). The

focus should be on application development risk, the associated business risk and technical vulnerabilities, and how these could materialize and be controlled/addressed. Some controls will look the same for all application development activity, but many will need to reflect the way the development activity is taking place in the area under review.

With web-based application development, an XML language known as Simple Object Access Protocol (SOAP) is used to define APIs. SOAP will work with any OS and programming language that understands XML. SOAP is simpler than using the more complex RPC-based approach, with the advantage that modules are coupled loosely so that a change to one component does not normally require changes to other components.

The second key component of web development is the Web Services Description Language (WSDL), which is also based on XML. WSDL is used to identify the SOAP specification that is to be used for the code module API and the formats of the SOAP messages used for input and output to the code module. The WSDL is also used to identify the particular web service accessible via a corporate intranet or across the Internet by being published to a relevant intranet or Internet web server.

The final component of web services is another XML-based language—Universal Description, Discovery and Integration (UDDI). UDDI is used to make an entry in a UDDI directory, which acts as an electronic directory accessible via a corporate intranet or across the Internet and allows interested parties to learn of the existence of available web services.

Software Reengineering

Software reengineering is a process of updating an existing system by extracting and reusing design and program components. This process is used to support major changes in the way an organization operates. A number of tools are available to support this process. Typical methodologies used in software reengineering generally fall into the following categories:

- BPR is the thorough analysis and significant redesign of business processes and management systems to establish a better performing structure that is more responsive to the customer base and market conditions, while

yielding material cost savings.
- The service-oriented software reengineering methodology is based upon the service-oriented computer architecture, and the reengineering processes apply many concepts of RAD development leveraging responsible, accountable, consulted and informed (RACI) charts and UML modeling.

Reverse Engineering

Reverse engineering is the process of studying and analyzing an application, a software application or a product to see how it functions and to use that information to develop a similar system. This process can be carried out in different ways:

- Decompiling object or executable code into source code and using it to analyze the program
- Black-box-testing the application to be reverse-engineered to unveil its functionality

The major advantages of reverse engineering are:

- Faster development and reduced SDLC duration
- The possibility of introducing improvements by overcoming the reverse-engineered application drawbacks An IS auditor should be aware of the following risk items:
- Software license agreements often contain clauses prohibiting the licensee from reverse engineering the software so that no trade secrets or programming techniques are compromised.
- Decompilers are relatively new tools with functions that depend on specific computers, OSs and programming languages. Any change in one of these components may require developing or purchasing a new decompiler.

DevOps

DevOps refers to the integration of development and operations processes to eliminate conflicts and barriers. This integration can create a great deal of benefits, but it can also create new risk. Decisions to adopt DevOps should be made based on factors such as an organization’s climate, risk tolerance and culture and on the scope of the development project. Because DevOps changes the environment and often impacts an organization’s control environment and accepted level of risk, an IS auditor should ensure that there

is a proper separation of duties.

DevOps combines the concepts of agile development, agile infrastructure and flexible operations. It requires a bridge of communication between software development and operations and the application of agile principles to all functions that support the software development life cycle. The adoption of DevOps—and, ultimately, DevSecOps in some organizations—is often closely associated with the adoption of agile. DevSecOps uses two distinctive concepts: (1) the confluence of software development, information security and IT operations groups and (2) the use of automation in those activities.

Implementing DevOps processes can be done in a logical and systematic manner and used to enhance the maturity of software development.

An organization should consider the following controls when embracing a DevOps development approach:

- Automated software scanning
- Automated vulnerability scanning
- Web application firewall
- Developer application security training
- Software dependency management
- Access and activity logging
- Documented policies and procedures
- Application performance management
- Asset management and inventorying
- Continuous auditing and/or monitoring

Business Process Reengineering and Process Change

In a generic process, some form of information enters the process, is processed, and the outcome is measured against the goal or objective of the process. The level of detail needed depends highly on the complexity of the process, the knowledge of the affected staff and the company’s requirements regarding audit functionality (performance and compliance) of the process and shall fit into an existing quality management system.

Any output produced by a process must be bound to a business objective and

adhere to defined corporate standards. Monitoring of effectiveness (goal achievement), efficiency (minimum effort) and compliance must be done on a regular basis and should be included in management reports for review under the plan-do-check-act (PDCA) cycle.

BPR is the process of responding to competitive and economic pressures and customer demands to survive in the current business environment. This is usually done by automating system processes so that there are fewer manual interventions and manual controls. BPR achieved with the help of implementing an ERP system is often referred to as package-enabled reengineering (PER). Advantages of BPR are usually experienced when the reengineering process appropriately suits the business needs. BPR has increased in popularity as a method for achieving the goal of cost savings through streamlining operations.

The steps in a successful BPR are to:
- Define the areas to be reviewed.
- Develop a project plan.
- Gain an understanding of the process under review.
- Redesign and streamline the process.
- Implement and monitor the new process.
- Establish a continuous improvement process.

As a reengineering process takes hold, new results begin to emerge such as:
- New business priorities based on value and customer requirements
- A concentration on process as a means of improving product, service and profitability
- New approaches to organizing and motivating people inside and outside the enterprise
- New approaches to the use of technologies in developing, producing and delivering goods and services
- New approaches to the use of information as well as powerful and more accessible information technologies
- Refined roles for suppliers including outsourcing, joint development, quick response, just-in-time inventory and support
- Redefined roles for clients and customers, providing them with more direct

and active participation in the enterprise’s business process

A successful BPR/process change project requires the project team to perform the following for the existing processes:

- Process decomposition to the lowest level required for effectively assessing a business process (typically referred to as an elementary process), which is a unit of work performed with a definitive input and output
- Identification of customers, process-based managers or process owners responsible for processes from beginning to end
- Documentation of the elementary process-related profile information including:
  - Duration
  - Trigger (which triggers the process to act)
  - Frequency
  - Effort
  - Responsibility (process owner)
  - Input and output
  - External interfaces
  - System interaction
  - Risk and control information
  - Performance measurement information
  - Identified problematic areas and their root causes

The existing baseline processes must be documented—preferably in the form of flowcharts and related profile documents—so they can be compared to the processes after reengineering. The newly designed business processes inevitably involve changes in the way(s) of doing business and could impact the finances, philosophy and personnel of the organization; its business partners; and its customers.

Throughout the change process, the BPR team must be sensitive to organization culture, structure, direction and the components of change. Management must also be able to predict and/or anticipate issues and problems and offer appropriate resolutions that will accelerate the change process.

BPR teams can be used to facilitate and assist the staff in transitioning into the reengineered business processes. BPR professionals are valuable in monitoring progress toward the achievement of the strategic plan of the organization.

A major concern in BPR is that key controls may be reengineered out of a business process. An IS auditor's responsibility is to identify the existing key controls and evaluate the impact of removing these controls. If the controls are key preventive controls, an IS auditor must ensure that management is aware of the removal of the control and is willing to accept the potential material risk of not having that preventive control.

Benchmarking Process

Benchmarking is about improving business processes. It is defined as a continuous, systematic process for evaluating the products, services or work processes of organizations recognized as world-class "references" in a globalized world. Reference products, services or processes are systematically analyzed for one or more of the following purposes:

- Comparing and ranking
- Strategic planning; strengths, weaknesses, opportunities and threats (SWOT) analysis
- Investment decisions, company takeovers, mergers
- Product or process design or redesign/reengineering
- BPR

The following steps are conducted in a benchmarking exercise:

1. **Plan**—The benchmarking team identifies the critical processes and gains an understanding of how they are measured, the kinds of data that are needed and how the data need to be collected.
2. **Research**—The team collects baseline data about the processes of its own organization before collecting the data about other organizations. The next step is to identify the reference products or companies through sources such as business newspapers and magazines, quality award winners, trade journals, and consultants. Depending on the team's own preferences and resources, and on the marketplace, several scenarios may result:
   – Benchmarks that satisfy the organization's interest already exist at no

charge from professional associations, journals or analysis firms.
  - The organization may join or promote a survey launched by a single or multi-industry specialized web portal (e.g., a bookmark portal).
  - The organization may conduct or subcontract business intelligence.
  - The organization may enter into an agreement with one or more "benchmark partners" who agree to share information.

Note: Based on the information gathered during the research phase, steps 3 through 6 may be skipped or adapted.

3. **Observe**—The next step is to collect data and visit the benchmarking partner. There should be an agreement with the partner organization, a data collection plan and a method to facilitate proper observation.
4. **Analyze**—This step involves summarizing and interpreting the data collected and analyzing the gaps between an organization's process and its partner's process. Converting key findings into new operational goals is the goal of this stage.
5. **Adopt**—Adopting the results of benchmarking can be the most difficult step. In this step, the team needs to translate the findings into a few core principles and work down from principles to strategies to action plans.
6. **Improve**—Continuous improvement is the key focus in a benchmarking exercise. Benchmarking links each process in an organization with an improvement strategy and organizational goals.

IS Auditor's Role in Business Process Reengineering

When reviewing an organization's BPR efforts, an IS auditor must determine whether:

- The organization's change efforts are consistent with the overall culture and strategic plan of the organization.
- The reengineering team is trying to minimize any negative impact the change might have on the organization's staff.
- The BPR team has documented lessons to be learned after the completion of the BPR/process change project.

An IS auditor would also provide a statement of assurance or conclusion with

respect to the objectives of the audit.

3.3.6 SYSTEM DEVELOPMENT TOOLS AND PRODUCTIVITY AIDS

System development tools and productivity aids include CASE applications, code generators and 4GLs.

Computer-aided Software Engineering

Application development efforts require collecting, organizing and presenting a substantial amount of data at the application, systems and program levels. A substantial amount of the application development effort involves translating this information into program logic and code for subsequent testing, modification and implementation. This often is a time-consuming process but it is necessary to develop, use and maintain computer applications.

CASE is the use of automated tools to aid in the software development process. Their use may include the application of software tools for software requirements capture and analysis, software design, code production, testing, document generation, and other software development activities.

CASE products are generally divided into three categories:

1. **Upper CASE**—Products used to describe and document business and application requirements. This information includes data object definitions and relationships, and process definitions and relationships.
2. **Middle CASE**—Products used for developing the detailed designs. These include screen and report layouts, editing criteria, data object organization and process flows. When elements or relationships change in the design, it is necessary to make only minor alterations to the automated design and all other relationships are automatically updated.
3. **Lower CASE**—Products involved with the generation of program code and database definitions. These products use detailed design information, programming rules and database syntax rules to generate program logic, data file formats or entire applications.

Some CASE products cover two of these categories or all three of them.

CASE tools provide a uniform approach to system development, facilitate storage and retrieval of documents, and reduce the manual effort in developing and presenting system design information. This power of automation changes the nature of the development process by eliminating or combining some steps and altering the means of verifying specifications and applications.

An IS auditor needs to recognize the changes in the development process brought on by CASE. Some CASE systems allow a project team to produce a complete system from the DFDs and data elements without any traditional source code. In these situations, the DFDs and data elements become the source code.

An IS auditor should gain assurance that approvals are obtained for the appropriate specifications, users continue to be involved in the development process, and investments in CASE tools yield benefits in quality and speed. Other key issues that an IS auditor needs to consider with CASE include the following:

- CASE tools help in the application design process but do not ensure that the design, programs and system are correct or that they fully meet the needs of the organization.
- CASE tools should complement and fit into the application development methodology, but a project methodology needs to be in place for CASE to be effective. It should be understood and used effectively by the organization’s software developers.
- The integrity of data moved between CASE products or between manual and CASE processes needs to be monitored and controlled.
- Changes to the application should be reflected in stored CASE product data.
- Just like a traditional application, application controls need to be designed.
- The CASE repository (the database that stores and organizes the documentation, models and other outputs from the different phases) needs to be secured on a need-to-know basis. Strict version control should be maintained on this database.

An IS auditor may use CASE tools as several features facilitate the audit

process. DFDs may be used as an alternative to other flowcharting techniques. In addition, CASE tools can be used to develop interrogation software and EAMs. Repository reports should be used to gain an understanding of the system and review controls over the development process.

Code Generators

Code generators are tools that are often incorporated with CASE products, which generate program code based on parameters defined by a systems analyst or on data/entity flow diagrams developed by the design module of a CASE product. These products allow most developers to implement software programs with efficiency. An IS auditor should be aware of source code generated by such tools.

Fourth-generation Languages

4GLs are used in software development to reduce the overall effort and cost. The common characteristics of 4GLs are:

- **Nonprocedural language**—Most 4GLs do not obey the procedural paradigm of continuous statement execution and subroutine call and control structures. Instead, they are event-driven and make extensive use of object-oriented programming concepts such as objects, properties and methods.
  - For example, a COBOL programmer who wants to produce a report sorted in a given sequence must first open and read the data file, then sort the file and finally produce the report. A typical 4GL treats the report as an object with properties, such as input file name and sort order, and methods, such as sort file and print report.
  - Care should be taken when using 4GLs. Unlike traditional languages, 4GLs can lack the lower-level detail commands necessary to perform certain types of data-intensive or online operations. These operations are usually required when developing major applications. For this reason, the use of 4GLs as development languages should be weighed carefully against traditional languages already discussed.
- **Environmental independence (portability)**—Many 4GLs are portable across computer architectures, OSs and telecommunications monitors. Some 4GLs have been implemented on mainframe processors and

microcomputers.

- **Software facilities**—These facilities include the ability to design or paint retrieval screen formats, develop computer-aided training routines or help screens, and produce graphical outputs.
- **Programmer workbench concepts**—The programmer has access through the terminal to easy filing facilities, temporary storage, text editing and OS commands. This type of a workbench approach is closely associated with the CASE application development approach. It is often referred to as an IDE.
- **Simple language subsets**—4GLs generally have simple language subsets that can be used by less-skilled users in an information center.

4GLs are often classified in the following ways:

- **Query and report generators**—These specialized languages can extract and produce reports (audit software). Recently, more powerful languages have been produced that can access database records, produce complex online outputs and be developed in an almost natural language.
- **Embedded database 4GLs**—These depend on self-contained database management systems. This characteristic often makes them more user-friendly but also may lead to applications that are not integrated well with other production applications. Examples include FOCUS, RAMIS II and NOMAD 2.
- **Relational database 4GLs**—These high-level language products are usually an optional feature on a vendor's DBMS product line. These allow the applications developer to make better use of the DBMS product, but they often are not end-user-oriented. Examples include SQL+, MANTIS and NATURAL.
- **Application generators**—These development tools generate lower-level programming languages (i.e., 3GLs) such as COBOL and C. The application can be further tailored and customized. Data processing development personnel, not end users, use application generators.

3.3.7 INFRASTRUCTURE DEVELOPMENT/ACQUISITION PRACTICES

The physical architecture analysis, the definition of a new one and the

necessary road map to move from one to the other are critical tasks for an IT department. Their impact is not only economic but also technological because it decides many other choices downstream, such as operational procedures, training needs, installation issues and TCO.

Conflicting requirements—such as evolving toward a services-based architecture, legacy hardware considerations, secure data access independent of data location, zero data loss and 24/7 availability—ensure that no single platform satisfies all these requirements equally. Thus, physical architecture analysis cannot be based solely on price or isolated features. A formal, reasoned choice must be made.

Information and communication technologies (ICT) departments are often tasked with making these decisions. The suggested solution must accomplish the following:

- Ensure alignment of the ICT with corporate standards.
- Provide appropriate levels of security.
- Integrate with current IT systems.
- Consider IT industry trends.
- Provide future operational flexibility to support business processes.
- Allow for projected growth in infrastructure without major upgrades.
- Include technical architecture considerations for information security, secure storage, etc.
- Ensure cost-effective, day-to-day operational support.
- Foster the usage of standardized hardware and software.
- Maximize ROI, cost transparency and operational efficiency.

Project Phases of Physical Architecture Analysis

Figure 3.12 shows the project phases to physical architecture analysis and the time at which the vendor selection process may take place.

Review of Existing Architecture

To start the process, the most current documents describing the existing architecture must be reviewed. Participants of the first workshop will be specialists of the ICT department in all areas directly impacted by physical architecture. Examples are server, storage, security and overall IT

infrastructure.

Special care must be taken in characterizing all the operational constraints that impact physical architecture such as:

- Ground issues
- Size limits
- Weight limits
- Current power supply
- Environmental operating limitations (temperature and humidity minimum and maximum)
- Physical security issues

The output of the first workshop is a list of components of the current infrastructure and constraints defining the target physical architecture.

Analysis and Design

After reviewing the existing architecture, the analysis and design of the actual physical architecture has to be undertaken, adhering to good practices and meeting business requirements.

Draft Functional Requirements

With the first physical architecture design in hand, the first (draft) of functional requirements is composed. This material is the input for the next step and the vendor selection process.

Vendor and Product Selection

While the draft functional requirements are written, the vendor selection process proceeds in parallel.

Writing Functional Requirements

After finishing the draft functional requirements and feeding the second part of this project, the functional requirements document is written and will be introduced at the second architecture workshop with staff from all affected parties. The results will be discussed and a list of the requirements that need to be refined or added will be composed.

This is the last checkpoint before the sizing and the proof of concept (POC) starts, although the planning of the POC starts after the second workshop. With the finished functional requirements, the POC phase begins.

Proof of Concept

Establishing a POC is highly recommended to prove that the selected hardware, software and data are able to meet all expectations, including security requirements. The deliverable of the POC should be a running prototype, including the associated document and test protocols describing the tests and their results.

To start, the POC should be based on the results of the procurement phase described below in this section. For this purpose, a representative subset of the target hardware is used. The software to run the POC can be either test versions or software already supplied by the vendor; therefore, additional costs are expected to be minimal. To keep costs low, most elements of the framework are implemented in a simplified form. They will be extended to their final form in later phases.

The prototype should demonstrate the following features:

- The basic setup of the core security infrastructure
- Correct functionality of auditing components
- Basic but functional implementation of security measures as defined
- Secured transactions
- Characterization in terms of installation constraints and limits (server size, server current consumption, server weight, server room physical security)
- Performance
- Resiliency to include basic fail-over to a trusted operational state
- Funding and costing model
- Data and algorithm

Related implementation projects that prepare for deployment should also be part of the POC because they will be used in the same way as they are used in the production physical architecture. At the end of this phase, a last workshop is held where the production sizing and layout is adapted to include POC conclusions.

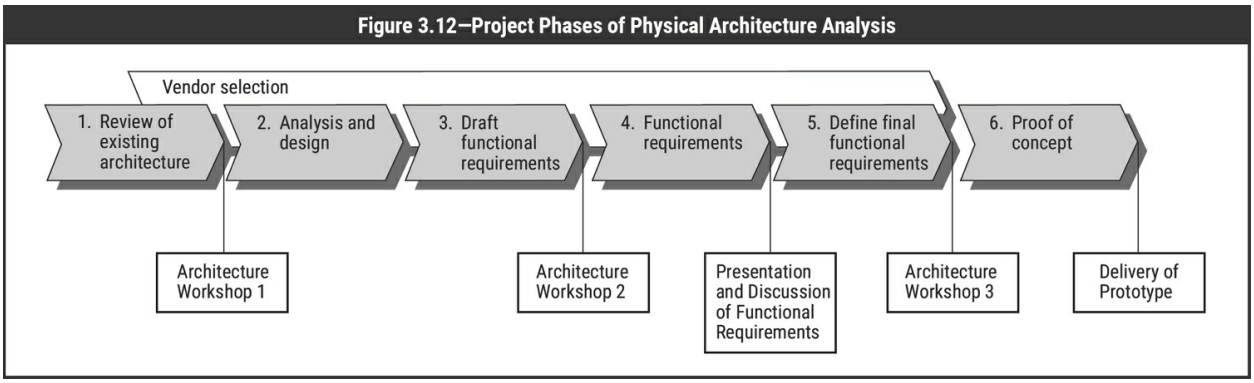


Figure 3.12—Project Phases of Physical Architecture Analysis

Additional considerations may apply if the entity goes in for an outsourcing/offshoring model for deployment and operation of applications. Also, the platform for operation of the IT environment (i.e., owned, cloud-based, virtualization) can give rise to additional considerations. For example, if the enterprise operates in a highly regulated industry or an industry that demands high levels of availability, adequate redundancy and safeguards for ensuring data privacy and confidentiality may have to be factored in while testing the POC.

Planning Implementation of Infrastructure

To ensure the quality of the results, it is necessary to use a phased approach to fit the entire puzzle together. It is also fundamental to set up the communication processes to other projects like those described earlier. Through these different phases the components are fit together, and a clear understanding of the available and contactable vendors is established by using the selection process during the procurement phase and beyond. Furthermore, it is necessary to select the scope of key business and technical requirements to prepare the next steps, which include the development of the delivery, installation and test plans. Moreover, to ensure a future proven solution, it is crucial to choose the right partners with the right skills.

As shown in **figure 3.13**, the requirements analysis is not part of this process but constantly feeds results into the process. If a Gantt chart is produced with these phases, most likely some phases overlap; therefore, the different phases must be considered an iterative process.

During the four different phases, it is necessary to fit all the components

together to prepare for projects downstream (e.g., data migration).

Procurement Phase

During the procurement phase, communication between the business and the analysis project is established to provide an overview of the chosen solution and determine the quantity structure of the deliverables. The requirements statements are also produced. Additionally, the procurement process begins the service-level management process. During these activities, the preferred partners are invited to the negotiations process and the deliverables, contracts and SLAs are signed (**figure 3.14**).

Delivery Time

During the delivery time phase, the delivery plan is developed (**figure 3.15**). This phase overlaps in some parts with the procurement phase.

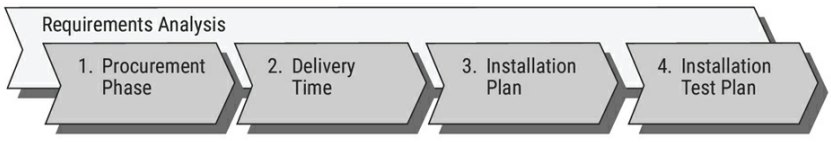
The delivery plan should include topics such as priorities, goals and nongoals, key facts, principles, communication strategies, key indicators, and progress on key tasks and responsibilities.

Installation Plan

During the installation planning phase, the installation plan is developed in cooperation with all affected parties (**figure 3.16**).

An additional step is to review the plan with the involved parties and, of course, with those responsible for the integration projects. This is an iterative process.

Figure 3.13—Project Phases of Planning the Implementation of Infrastructure



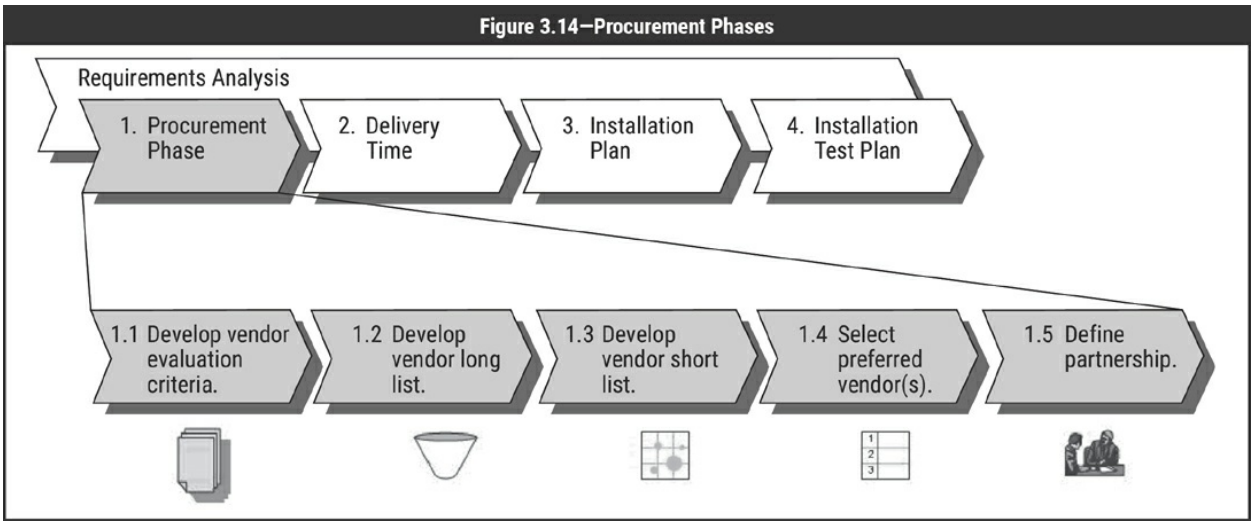


Figure 3.14—Procurement Phases

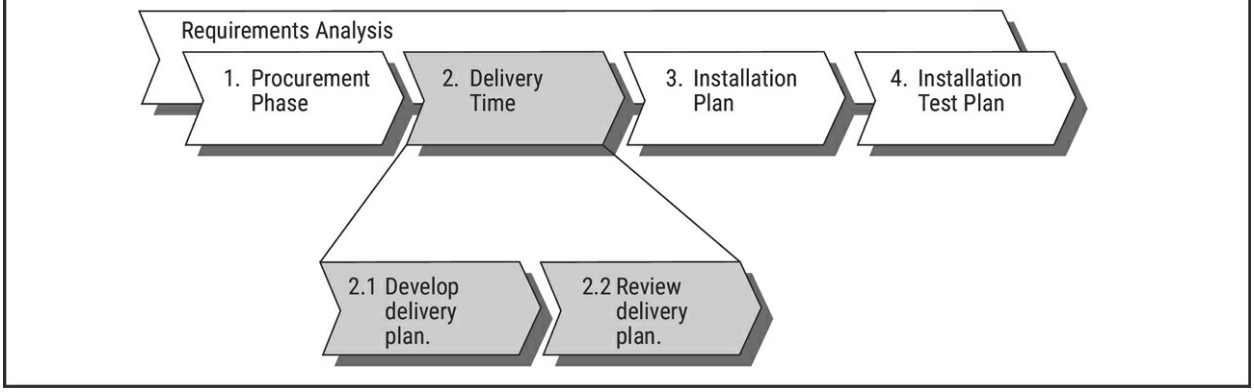


Figure 3.15—Delivery Time

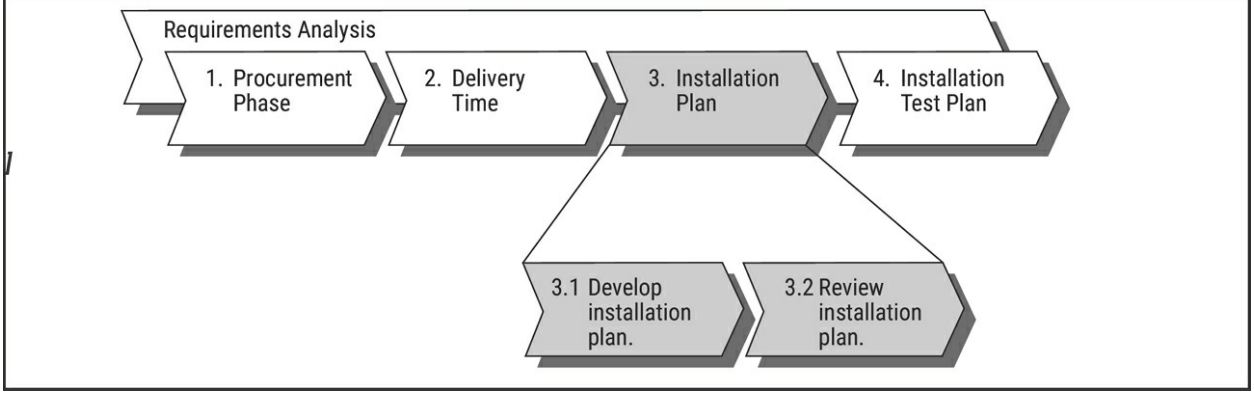


Figure 3.16—Installation Plan

Installation Test Plan

Based on the known dependencies of the installation plan, the test plan is developed (**figure 3.16**).

The test plan includes test cases, basic requirements' specifications, definition of the processes, and, as far as possible, measurement information for the applications and the infrastructure. Part one of the project (analysis of the physical architecture) must be completed, and the needed infrastructure decisions must be made.

3.3.8 HARDWARE/SOFTWARE ACQUISITION

Selection of a computer hardware and software environment frequently requires the preparation of specifications for distribution to hardware/software (HW/SW) vendors and criteria for evaluating vendor proposals. The specifications are sometimes presented to vendors in the form of an invitation to tender (ITT), also known as a request for proposal (RFP). The specifications must define, as completely as possible, the usage, tasks and requirements for the equipment needed and must include a description of the environment in which that equipment will be used.

When acquiring a system, the specifications should include the following:

- Organizational descriptions indicating whether the computer facilities are centralized or decentralized, distributed, outsourced, manned or lights-out
- HW/SW evaluation assurance levels (EALs) for security robustness
- Information processing requirements such as:
  - Major existing application systems and future application systems
  - Workload and performance requirements
  - Processing approaches (e.g., online/batch, client-server, real-time databases, continuous operation)
- Hardware requirements such as:
  - CPU speed
  - Disk space requirements
  - Memory requirements
  - Number of CPUs required
  - Peripheral devices (e.g., sequential devices such as tape drives; direct access devices such as magnetic disk drives, printers, compact disc drives, digital video disc drives, universal serial bus [USB] peripherals and secure digital multimedia cards [SD/MMC]) required or to be excluded (usually for security reasons)

  - Data preparation/input devices that accept and convert data for machine processing
  - Direct entry devices (e.g., terminals, point-of-sale terminals or automated teller machines)
  - Networking capability (e.g., Ethernet connections, modems and integrated services digital network [ISDN] connections)
  - Number of terminals or nodes the system needs to support
- System software applications such as:
  - OS software (current version and any required upgrades)
  - Utilities
  - Compilers
  - Program library software
  - Database management software and programs
  - Communications software
  - Access control software
  - Job scheduling software
- Support requirements such as:
  - System maintenance (for preventive, detective [fault reporting] or corrective purposes)
  - Training (user and technical staff)
  - Backups (daily and disaster backups)
  - Patching
- Adaptability requirements such as:
  - Hardware and software upgrade capabilities
  - Compatibility with existing hardware and software platforms
  - Changeover to other equipment capabilities
- Constraints such as:
  - Staffing levels
  - Existing hardware capacity
  - Delivery dates
- Conversion requirements such as:
  - Test time for the hardware and software
  - System conversion facilities
  - Cost/pricing schedule

Acquisition Steps

When purchasing (acquiring) HW/SW from a vendor, consideration should be given to the following:

- Testimonials or visits with other users
- Provisions for competitive bidding
- Analysis of bids against requirements
- Comparison of bids against each other using predefined evaluation criteria
- Analysis of the vendor's financial condition
- Analysis of the vendor's capability to provide maintenance and support (including training)
- Review of delivery schedules against requirements
- Pedigree of the hardware to verify it is not sourced from "gray market" supply sources (through distribution sources that are legal but are unofficial, unauthorized or unintended by the original manufacturer) that can increase the risk of malware and other unknown operability of the product
- Analysis of hardware and software upgrade capability
- Analysis of security and control facilities
- Evaluation of performance against requirements
- Review and negotiation of price
- Review of contract terms (including warranties, penalties and right-to-audit clauses)
- Preparation of a formal written report summarizing the analysis for each of the alternatives and justifying the selection based on benefits and cost

The criteria and data used for evaluating vendor proposals should be properly planned and documented. The following are some of the criteria that should be considered in the evaluation process:

- **Turnaround time**—The time that the help desk or vendor takes to fix a problem from the moment it is logged in
- **Response time**—The time a system takes to respond to a specific query by the user
- **System reaction time**—The time taken for logging into a system or getting connected to a network
- **Throughput**—The quantity of useful work made by the system per unit of time. Throughput can be measured in instructions per second or some other unit of performance. When referring to a data transfer operation,

throughput measures the useful data transfer rate and is expressed in kilobits per second (Kbps), megabits per second (Mbps) and gigabits per second (Gbps).

- **Workload**—The capacity to handle the required volume of work or the volume of work that the vendor's system can handle in a given time frame
- **Compatibility**—The capability of an existing application to run successfully on the newer system supplied by the vendor
- **Capacity**—The capability of the newer system to handle a number of simultaneous requests from the network for the application and the volume of data that it can handle from each of the users
- **Utilization**—The system availability time versus the system downtime

IS Auditor's Role in Hardware Acquisition

When performing an audit of this area, an IS auditor should:

- Determine if the acquisition process began with a business need and whether the hardware requirements for this need were considered in the specifications.
- Determine if several vendors were considered and whether the comparison between them was done according to the aforementioned criteria.

3.3.9 SYSTEM SOFTWARE ACQUISITION

Every time a technological development has allowed for increased computing speeds or new capabilities, these have been absorbed immediately by the demands placed on computing resources by more ambitious applications. Consequently, improvements have led to decentralized, interconnected open systems through functions bundled in OS software to meet these needs. For example, network management and connectivity are features now found in most OSs.

It is IS management's responsibility to be aware of HW/SW capabilities because they may improve business processes and provide expanded application services to businesses and customers in a more effective way. Short- and long-term plans should document IS management's plan for migrating to newer, more efficient and more effective OSs and related systems software.

When selecting new system software, a number of business and technical issues must be considered including:

- Business, functional and technical needs and specifications
- Cost and benefit(s)
- Obsolescence
- Compatibility with existing systems
- Security
- Demands on existing staff
- Training and hiring requirements
- Future growth needs
- Impact on system and network performance
- Open source code versus proprietary code

The feasibility study should contain documentation that supports the decision to acquire the software. Depending on the software required there could be four cases:

1. Software is required for a generic business process for which vendors are available and software can be implemented without customization.
2. Software (the vendor's) needs to be customized to suit business processes.
3. Software needs to be developed by the vendor.
4. Software is available as a service through the cloud (software as a service [SaaS]). This is generally available for generic processes.

A project team with participation by technical support staff and key users should be created to write an RFP or ITT. An RFP needs to be prepared separately for each case mentioned previously. The invitation to respond to an RFP should be widely distributed to appropriate vendors and, if possible, posted via a public procurement medium (Internet or newspaper). This process allows the business to determine which of the responding vendors' products offers the best solution at the most cost-effective price.

The RFP should include the areas shown in **figure 3.17**.

When the product and related services are known in advance, a user organization often prefers an ITT so it can obtain the best combination of price and services. This is more applicable when procurement of hardware,

network, database, etc., is involved. When the requirement is more toward a solution and related support and maintenance, an organization generally prefers an RFP, so the capability, experience and approach can be measured against the requirement. This is more applicable in system integration projects such as ERP and supply chain management (SCM) that involve delivery or escrowing of source code.

Figure 3.17—RFP Contents	
Item	**Description**
Product versus system requirements	The chosen vendor's product should come as close as possible to meeting the defined requirements of the system. If no vendor's product meets all of the defined requirements, the project team, especially the users, will have to decide whether to accept the deficiencies. An alternative to living with a product's deficiencies is for the vendor or the purchaser to make customized changes to the product.
Product scalability and interoperability	The project management should not only look at vendor's product ability to meet the existing requirements for the project but also the ability of the product to grow and/or contract with the organization's business processes. Vendor products should be assessed as to the applications' ability to interconnect with other systems whose interconnections are currently out of the project's scope but may be needed in the future.
Customer references	Project management should check vendor-supplied references to validate the vendor's claims of product performance and completion of work by the vendor.
Vendor viability/financial stability	The vendor supplying or supporting the product should be reputable and able to provide evidence of financial stability. A vendor may not be able to prove financial stability; if the product is new, the vendor presents a substantially higher risk to the organization.
Availability of complete and reliable documentation	The vendor should be willing and able to provide a complete set of system documentation for review prior to acquisition. The level of detail and precision found in the documentation may be an indicator of the detail and precision utilized within the design and programming of the system itself.
Vendor support	The vendor should have available a complete line of support products for the software package. This may include a 24-hour, seven-day-a-week help line, onsite training during implementation, product upgrades, automatic new version notification and onsite maintenance, if requested.

Source code availability	The source code should be received either from the vendor initially or there should be provisions for acquiring the source code in the event that the vendor goes out of business. Usually, these clauses are part of a software escrow agreement in which a third party holds the software in escrow should such an event occur. The acquiring company should ensure that product updates and program fixes are included in the escrow agreement.
Number of years of experience in offering the product	More years indicate stability and familiarity with the business that the product supports.
A list of recent or planned enhancements to the product, with dates	A short list suggests the product is not being kept current.
Number of client sites using the product with a list of current users	A larger number suggests wide acceptance of the product in the marketplace.
Acceptance testing of the product	Such testing is crucial in determining whether the product really satisfies the system requirements. This is allowed before a purchasing commitment must be made.

Often, prior to the development of an RFP, an organization will develop a request for information (RFI) to solicit software development vendors for advice in addressing problems with existing systems. Information obtained in this manner may be used to develop an RFP. The project team needs to carefully examine and compare the vendors' responses to the RFP. This comparison should be done using an objective method such as a scoring and ranking methodology. After the RFP responses have been examined, the project team may be able to identify a single vendor whose product satisfies most or all of the stated requirements in the RFP. Other times, the team may narrow the list to two or three acceptable candidates (i.e., short list of vendors). In evaluating the best-fit solution and vendor against the given set of business requirements and conditions, a suitable methodology of evaluation should be adopted. The methodology should ensure objective, equitable and fair comparison of the products/vendors (e.g., a gap analysis to find out the differences between requirements and software, the parameters required to modify).

It is important to keep in mind the minimum and recommended requirements to use software, including:

- Required hardware such as memory, disk space, and server or client characteristics
- OS versions and patch levels supported
- Additional tools such as import and export tools
- Databases supported

In addition, it is likely that more than one product/vendor fits the requirements, with advantages and disadvantages with respect to each other. To resolve such a situation, agenda-based presentations should be requested from the short-listed vendors. The agenda-based presentations are scripted business scenarios that are designed to show how the vendor will perform certain critical business functions. Vendors are typically invited to demonstrate their product and follow the sample business scenarios given to them to prepare. It is highly recommended to include adequate participation from various user groups when evaluating the product's/vendor's fit and the system's ease of use. The project team thus has an opportunity to check the intangible issues such as the vendor's knowledge of the product and the vendor's ability to understand the business issue at hand. Having each short-listed vendor demonstrate its product following a scripted document also enables the project team to evaluate and finalize the product/vendor selection with knowledge and objectivity built into the process. The finalist vendor candidate is then requested to organize site visits to confirm the findings from the agenda-based presentations and check the system in a live environment. Once the finalist is confirmed, a conference room pilot needs to be conducted. A conference room pilot enables the project team to understand the system with a hands-on session with business end users and identify the areas that need certain customizations or workarounds.

Additionally, for the short list of vendors, it can be beneficial for the project team to talk to current users of each of the potential products. If it can be arranged and cost-justified, an onsite visit can be even more beneficial. Whenever possible, the organizations chosen should be those that use the products in a manner that is similar to the way the organization selecting the products plans to use them.

An IS auditor should encourage the project team to contact current users. The

information obtained from these discussions or visits validates statements made in the vendor's proposal and can determine which vendor is selected. The discussions with the current users should concentrate on each vendor's:

- **Reliability**—Are the vendor's deliverables (enhancements or fixes) dependable?
- **Commitment to service**—Is the vendor responsive to problems with its product? Does the vendor deliver on time?
- **Commitment to providing training, technical support and documentation for its product**—What is the level of customer satisfaction?

Upon completing the activities cited, vendor presentations and final evaluations, the project team can make a product/vendor selection. The reasons for making a particular choice should be documented.

The last step in the acquisition process is to negotiate and sign a contract for the chosen product. Appropriate legal counsel should review the contract prior to its signing. The contract should contain the following items:

- Specific description of deliverables and their costs
- Commitment dates for deliverables
- Commitments for delivery of documentation, fixes, upgrades, new release notifications and training
- Commitments for data migration
- Allowance for a software escrow agreement, if the deliverables do not include source code
- Description of the support to be provided during installation/customization
- Criteria for user acceptance
- Provision for a reasonable acceptance testing period, before the commitment to purchase is made
- Allowance for changes to be made by the purchasing company
- Maintenance agreement
- Allowance for copying software for use in business continuity efforts and for test purposes
- Payment schedule linked to actual delivery dates
- Confidentiality clauses
- Data protection clauses

Managing the contract should also involve a major level of effort to ensure that deployment efforts are controlled, measured and improved on, where appropriate. This may include regular status reporting requirements. Additionally, the milestones and metrics to be reported against should be agreed with the vendor.

Integrated Resource Management Systems

An integrated solutions implementation is a very large software acquisition project. The acquisition and implementation of an ERP system impacts the way that an organization does business and its entire control environment, technological direction and internal resources. Generally, an organization that adopts an integrated solution is required to convert management philosophies, policies and practices to those of the integrated software solution providers, notwithstanding the numerous customization options. In this respect, such a solution will either impair or enhance IT's ability to support the organization's mission and goals. When considering a change of this magnitude, it is imperative that a thorough impact and risk assessment be conducted.

When implementing an ERP solution or any off-the-shelf software, the business unit has the option of implementing and configuring the new system in the simplest configuration possible: as-is, out-of-the-box, and not developing any additional functionality or customization to bridge the gaps in an organization's specific business processes. The business opts to change business processes to suit the industry standard as dictated by the software solution.

While this decision results in less software design, development and testing work than does customization, it does require greater change in the business units to work differently. Due to the large costs in software development, maintenance, and continuous upgrading and patching, customization is not usually recommended by software vendors.

Because of the magnitude of the risk involved, it is imperative that senior management assess and approve all plans and changes in the system's architecture, technological direction, migration strategies and IS budgets.

IS Auditor's Role in Software Acquisition

An IS auditor should be involved in the software acquisition process to determine whether an adequate level of security controls has been considered prior to any agreement being reached. If security controls are not part of the software, it may become difficult to ensure data integrity for the information that will be processed through the system. Risk involved with the software package includes inadequate audit trails, password controls and overall security of the application. Because of the risk, an IS auditor should ensure that these controls are built into the software application.

An IS auditor should perform the following when reviewing software acquisition:

- Analyze the documentation from the feasibility study to determine whether the decision to acquire a solution was appropriate (including consideration of common criteria evaluations).
- Review the RFP to ensure that it covers the items listed in this section.
- Determine whether the selected vendor is supported by RFP documentation.
- Attend agenda-based presentations and conference room pilots to ensure that the system matches the vendor's response to the RFP.
- Review the vendor contract prior to its signing to ensure that it includes the items listed.
- Ensure the contract is reviewed by legal counsel before it is signed.
- Review the RFP to ensure security responses are included by the vendor.

3.4 CONTROL IDENTIFICATION AND DESIGN

An IS auditor must be able to identify and understand controls designed to ensure the authorization, accuracy and completeness of data input to, processing by and output from various business and computer applications. An IS auditor must also be familiar with control techniques and how each may be evidenced in the form of reports, logs and audit trails.

3.4.1 INPUT/ORIGINATION CONTROLS

Input control procedures must ensure that every transaction to be processed is

entered, processed and recorded accurately and completely. These controls should ensure that only valid and authorized information is input and these transactions are processed only once. These include machine and manual inputs. In an integrated systems environment, output generated by one system is the input for another system. Therefore, the system receiving the output of another system as input/origination must, in turn, apply edit checks, validations and access controls to those data.

Input Authorization

Input authorization verifies that all transactions have been authorized and approved by management. Authorization of input helps ensure that only authorized data are entered for processing by applications. Authorization can be performed online when the data are entered into the system. A computer-generated report listing the items requiring manual authorization may also be generated. It is important that controls exist throughout processing to ensure that the authorized data remain unchanged. This can be accomplished through various accuracy and completeness checks incorporated into an application's design.

Types of authorization include:

- **Signatures on batch forms or source documents**—Provide evidence of proper authorization
- **Online access controls**—Ensure that only authorized individuals may access data or perform sensitive functions
- **Unique passwords**—Ensure that access authorization cannot be compromised through use of another individual's authorized data access. Individual unique passwords also provide accountability for data changes.
- **Terminal or client workstation identification**—Limits input to specific terminals or workstations as well as to individuals. Terminals or client workstations in a network can be configured with a unique form of identification such as serial number or computer name that is authenticated by the system.
- **Source documents**—Record the data. A source document may be a piece of paper, a turnaround document or an image displayed for online data input. A well-designed source document achieves several purposes. It increases the speed and accuracy with which data can be recorded, controls

work flow, facilitates preparation of the data in machine-readable form for pattern recognition devices, increases the speed and accuracy with which data can be read, and facilitates subsequent reference checking.
- **Input data validation**—Ensures that information is being received in the expected format and that there is no malicious or manipulative activity taking place with inputs

Ideally, source documents should be preprinted or electronic forms to provide consistency, accuracy and legibility. Source documents should include standard headings, titles, notes and instructions. Source document layouts should:
- Emphasize ease of use and readability.
- Group similar fields together to facilitate input.
- Provide predetermined input codes to reduce errors.
- Contain appropriate cross-reference numbers or a comparable identifier to facilitate research and tracing.
- Use boxes to prevent field size errors.
- Include an appropriate area for management to document authorization.

All source documents should be appropriately controlled. Procedures should be established to ensure that all source documents have been input and considered. Prenumbering source documents facilitates this control.

Batch Controls and Balancing

Batch controls group input transactions to provide control totals. The batch control can be based on total monetary amount, total items, total documents or hash totals:
- **Total monetary amount**—Verification that the total monetary value of items processed equals the total monetary value of the batch documents. For example, the total monetary value of the sales invoices in the batch agrees with the total monetary value of the sales invoices processed. This provides assurance on the completeness and accuracy of the sales value processed for the batch.
- **Total items**—Verification that the total number of items included on each document in the batch agrees with the total number of items processed. For example, the total number of units ordered in the batch of invoices agrees

with the total number of units processed. This provides assurance on the completeness and accuracy of the units ordered in the batch processed.
- **Total documents**—Verification that the total number of documents in the batch equals the total number of documents processed. For example, the total number of invoices in a batch agrees with the total number of invoices processed. This provides assurance on the completeness of the number of invoices processed.
- **Hash totals**—Verification that the total in a batch agrees with the total calculated by the system. Hash total is the total of nonvalue numeric fields in the batch (such as total of dates or customer number fields, which, by themselves, do not have informative value). This provides assurance on the completeness and accuracy of data entered for the numeric fields in the batch.

Batch header forms are a data preparation control. All input forms should be clearly identified with the application name and transaction codes. Batch balancing can be performed through manual or automated reconciliation. Batch totaling must be combined with adequate follow-up procedures. Adequate controls should exist to ensure that each transaction creates an input document, all documents are included in a batch, all batches are submitted for processing, all batches are accepted by the computer, batch reconciliation is performed, procedures for the investigation and timely correction of differences are followed, and controls exist over the resubmission of rejected items.

Types of batch balancing include:
- **Batch registers**—Enable recording of batch totals and subsequent comparison with system reported totals
- **Control accounts**—Control account use through an initial edit file to determine batch totals. The data are then processed to the master file, and a reconciliation is performed between the totals processed during the initial edit file and the master file.
- **Computer agreement**—Compares batch header details that record the batch totals to calculated totals, either accepting or rejecting the batch

Error Reporting and Handling

Input processing requires that controls be identified to verify that only correct data are accepted into the system and input errors are recognized and corrected.

Data conversion error corrections are needed during the data conversion process. Errors can occur due to duplication of transactions and inaccurate data entry. These errors can, in turn, impact the completeness and accuracy of the data. Corrections to data should be processed through normal data conversion processes and should be verified, authorized and reentered into the system as a part of the normal processing.

Input error handling can be processed by:

- **Rejecting only transactions with errors**—Only transactions containing errors would be rejected; the rest of the batch would be processed.
- **Rejecting the whole batch of transactions**—Any batch containing errors would be rejected for correction prior to processing.
- **Holding the batch in suspense**—Any batch containing errors would not be rejected; however, the batch would be held in suspense, pending correction.
- **Accepting the batch and flagging error transactions**—Any batch containing errors would be processed; however, those transactions containing errors would be flagged for identification, enabling subsequent error correction.

Input control techniques:

- **Transaction log**—Contains a detailed list of all updates. The log can be either manually maintained or provided through automatic computer logging. A transaction log can be reconciled to the number of source documents received to verify that all transactions have been input.
- **Reconciliation of data**—Controls whether all data received are properly recorded and processed
- **Documentation**—Records (written evidence) user, data entry and data control procedures
- **Error correction procedures**—Includes:
  - Logging of errors
  - Timely corrections

  - Upstream resubmission
  - Approval of corrections
  - Suspense file
  - Error file
  - Validity of corrections
- **Anticipation**—Anticipates (user or control group) the receipt of data
- **Transmittal log**—Documents transmission or receipt of data
- **Cancellation of source documents**—Procedures to cancel source documents such as by punching with holes or marking them to avoid duplicate entry
- **Input sanitization**—Checks user input prior to storing it in a database or using it for other purposes in order to prevent malicious code injection

3.4.2 PROCESSING PROCEDURES AND CONTROLS

Processing procedures and controls are meant to ensure the reliability of application program processing. An IS auditor should understand the procedures and controls that can be exercised over processing to evaluate what exposures are covered by these controls and what exposures remain.

Data Validation and Editing Procedures

Procedures should be established to ensure that input data are validated and edited as close to the time and point of origination as possible. Preprogrammed input formats ensure that data are input to the correct field in the correct format. If input procedures allow supervisor overrides of data validation and editing, automatic logging should occur. A manager who did not initiate the override should review this log. Data validation is meant to identify data errors, incomplete or missing data, and inconsistencies among related data items. Front-end data editing and validation can be performed if intelligent terminals are used.

Edit controls are preventive controls that are used in a program before data are processed. If not in place or not working effectively, the preventive controls are not effective. This may cause processing of inaccurate data. **Figure 3.18** describes various types of data validation edits.

Processing Controls

Processing controls are meant to ensure the completeness and accuracy of accumulated data. They ensure that data in a file/database remain complete and accurate until changed as a result of authorized processing or modification routines. The following are processing control techniques that can be used to address the issues of completeness and accuracy of accumulated data:

Figure 3.18—Data Validation Edits and Controls

Edits	Description
Sequence check	The control number follows sequentially and any sequence or duplicated control numbers are rejected or noted on an exception report for follow-up purposes. For example, invoices are numbered sequentially. The day's invoices begin with 12001 and end with 15045. If any invoice larger than 15045 is encountered during processing, that invoice would be rejected as an invalid invoice number.
Limit check	Data should not exceed a predetermined amount. For example, payroll checks should not exceed US $4,000. If a check exceeds US $4,000, the data would be rejected for further verification/authorization.
Range check	Data should be within a predetermined range of values. For example, product type codes range from 100 to 250. Any code outside this range should be rejected as an invalid product type.
Validity check	Programmed checking of the data validity in accordance with predetermined criteria. For example, a payroll record contains a field for marital status and the acceptable status codes are M or S. If any other code is entered, the record should be rejected.
Reasonableness check	Input data are matched to predetermined reasonable limits or occurrence rates. For example, a widget manufacturer usually receives orders for no more than 20 widgets. If an order for more than 20 widgets is received, the computer program should be designed to print the record with a warning indicating that the order appears unreasonable.
Table lookups	Input data comply with predetermined criteria maintained in a computerized table of possible values. For example, the input clerk enters a city code of 1 to 10. This number corresponds with a computerized table that matches the code to a city name.
Existence check	Data are entered correctly and agree with valid predetermined criteria. For example, a valid transaction code must be entered in the transaction code field.

Key verification	The keying process is repeated by a separate individual using a machine that compares the original keystrokes to the repeated keyed input. For example, the worker number is keyed twice and compared to verify the keying process.
Check digit	A numeric value that has been calculated mathematically is added to data to ensure that the original data have not been altered or an incorrect, but valid, value substituted. This control is effective in detecting transposition and transcription errors. For example, a check digit is added to an account number so it can be checked for accuracy when it is used.
Completeness check	A field should always contain data rather than zeros or blanks. A check of each byte of that field should be performed to determine that some form of data, not blanks or zeros, is present. For example, a worker number on a new employee record is left blank. This is identified as a key field and the record would be rejected, with a request that the field be completed before the record is accepted for processing.
Duplicate check	New transactions are matched to those previously input to ensure that they have not already been entered. For example, a vendor invoice number agrees with previously recorded invoices to ensure that the current order is not a duplicate and, therefore, the vendor will not be paid twice.
Logical relationship check	If a particular condition is true, then one or more additional conditions or data input relationships may be required to be true and consider the input valid. For example, the hire date of an employee may be required to be more than 16 years past his/her date of birth.

- **Manual recalculations**—Manually recalculation of a sample of transactions to ensure that processing is accomplishing the anticipated task
- **Editing**—A program instruction or subroutine that tests the accuracy, completeness and validity of data. It may be used to control input or later processing of data.
- **Run-to-run totals**—Verification of data values through the stages of application processing. Run-to-run total verification ensures that data read into the computer were accepted and then applied to the updating process.
- **Programmed controls**—Software that detects and initiates corrective action for errors in data and processing. For example, if the incorrect file or file version is provided for processing, the application program could display messages instructing that the proper file and version be used.
- **Reasonableness verification of calculated amounts**—An application

program that verifies the reasonableness of calculated amounts. The reasonableness can be tested to ensure appropriateness to predetermined criteria. Any transaction that is determined to be unreasonable may be rejected pending further review.

- **Limit checks on amounts**—Assurance provided, through the use of predetermined limits, that amounts have been keyed or calculated correctly. Any transaction exceeding the limit may be rejected for further investigation.
- **Reconciliation of file totals**—Should be performed on a routine basis. Reconciliations may be performed through the use of a manually maintained account, a file control record or an independent control file.
- **Exception reports**—Generated by a program that identifies transactions or data that appear to be incorrect. These items may be outside a predetermined range or may not conform to specified criteria.

Data File Control Procedures

File controls should ensure that only authorized processing occurs to stored data. Types of controls over data files are shown in **figure 3.19**. Contents of data files, or database tables, generally fall into four categories:

- **System control parameters**—The entries in these files change the workings of the system and may alter controls exercised by the system—for example, the tolerance allowed before an exceptional transaction is reported or blocked. Any change to these files should be controlled in a similar way to program changes.
- **Standing data**—These “master files” include data, such as supplier/customer names and addresses, that do not frequently change and are referred to during processing. These data should be authorized before entry or maintenance. Input controls may include a report of changed data that is checked and approved. Audit trails may log all changes.
- **Master data/balance data**—Running balances and totals that are updated by transactions should not be capable of adjustment except under strict approval and review controls. Audit trails are important here since there may be financial reporting implications for the change.
- **Transaction files**—These are controlled using validation checks, control totals, exception reports, etc.

It should be noted that the controls built into an application represent the management design of controls on how a business process should be run. While an application contains the rules for the business, the data that are the outcome of the processing are stored in the database. An entity may have the best controls built into the application, but if management personnel directly update data in the database, then the benefit of the best controls in the application will be overridden.

However, in some situations, organizations may have to carry out direct updates to a database. For example, if, due to a systems outage, the transactions could not be processed in real time for a few days, it is not practical to insist that once the system availability is restored, the backlog should be entered through the application (front end) before the transactions of the subsequent days are entered or processed. In such cases, management may decide to catch up on the backlog by directly updating the transactions in the database (back end).

An IS auditor should ensure that there are controls in place to ensure that such direct back-end data fixes are supported by authorization of the business for completeness and accuracy and are processed subject to computer operations controls. The important point to remember is that, in any organization, the quality of application controls is only as good as the quality of controls around direct back-end data fixes.

3.4.3 OUTPUT CONTROLS

Output controls provide assurance that the data delivered to users will be presented, formatted and delivered in a consistent and secure manner.

Output controls include:

- **Logging and storage of negotiable, sensitive and critical forms in a secure place**—Negotiable, sensitive or critical forms should be properly logged and secured to provide adequate safeguards against theft, damage or disclosure. The form log should be routinely reconciled to have inventory on hand, and any discrepancies should be properly researched.
- **Computer generation of negotiable instruments, forms and signatures**—The computer generation of negotiable instruments, forms and signatures

should be properly controlled. A detailed listing of generated forms should be compared to the physical forms received. One should properly account for all exceptions, rejections and mutilations.

- **Report accuracy, completeness and timeliness**—Often reports are generated using third-party data analysis and reporting applications (ESSbase, etc.). Even with the most reliable and accurate data sources, improperly configured, constructed and prepared reports are still a significant risk. Report design and generation specifications, templates and creation/change request processes are critical system output controls.
- **Reports generated from the system**—These represent the data that management relies upon for business decisions and review of business results. Therefore, ensuring the integrity of data in reports is key for the reliability of information in information systems. An IS auditor should validate that the reports are accurate and provide correct representation of the source data.

Figure 3.19—Data File Controls

Method	Description
Before and after image reporting	Computer data in a file prior to and after a transaction is processed can be recorded and reported. The before and after images make it possible to trace the impact transactions have on computer records.
Maintenance error reporting and handling	Control procedures should be in place to ensure that all error reports are properly reconciled and corrections are submitted on a timely basis. To ensure SoD, error corrections should be reviewed properly and authorized by personnel who did not initiate the transaction.
Source documentation retention	Source documentation should be retained for an adequate time period to enable retrieval, reconstruction or verification of data. Policies regarding the retention of source documentation should be enforced. Originating departments should maintain copies of source documentation and ensure that only authorized personnel have access. When appropriate, source documentation should be destroyed in a secure, controlled environment.
Internal and external labeling	Internal and external labeling of removable storage media is imperative to ensure that the proper data are loaded for processing. External labels provide the basic level of assurance that the correct data medium is loaded for processing. Internal labels, including file header records, provide assurance that the proper data files are used and allow for automated checking.

Version usage	It is critical that the proper version of a file be used as well as the correct file, for processing to be correct. For example, transactions should be applied to the most current database, while restart procedures should use earlier versions.
Data file security	Data file security controls prevent unauthorized access by unauthorized users that may have access to the application to alter data files. These controls do not provide assurances relating to the validity of data but ensure that unauthorized users who may have access to the application cannot alter stored data improperly.
One-for-one checking	Individual documents agree with a detailed listing of documents processed by the computer. It is necessary to ensure that all documents have been received for processing.
Prerecorded input	Certain information fields are preprinted on blank input forms to reduce initial input errors.
Transaction logs	All transaction input activity is recorded by the computer. A detailed listing, including date of input, time of input, user ID and terminal location, can then be generated to provide an audit trail. It also permits operations personnel to determine which transactions have been posted. This will help to decrease the research time needed to investigate exceptions and decrease recovery time if a system failure occurs.
File updating and maintenance authorization	Proper authorization for file updating and maintenance is necessary to ensure that stored data are safeguarded adequately, correct and up to date. Application programs may contain access restrictions in addition to the overall system access restrictions. The additional security may provide levels of authorization as well as an audit trail of file maintenance.
Parity checking	Data transfers in a computer system are expected to be made in a relatively error-free environment. However, when programs or vital data are transmitted, additional controls are needed. Transmission errors are controlled primarily by error-detecting or correcting codes. The former is used more often because error-correcting codes are costly to implement and are unable to correct all errors. Generally, error detection methods such as a check bit and redundant transmission are adequate. Redundancy checking is a common error-detection routine. A transmitted block of data containing one or more records or messages is checked for the number of characters or patterns of bits contained in it. If the numbers or patterns do not conform to predetermined parameters, the receiving device ignores the transmitted data and instructs the user to retransmit. Check bits are often added to the transmitted data by the telecommunications control unit and may be applied either horizontally or vertically. These checks are similar to the parity checks normally applied to data characters within on-premises equipment. A parity check on a single character generally is referred to as a vertical or column check, and a

	parity check on all the equivalent bits is known as a horizontal, longitudinal or row check. Use of both checks greatly improves the possibilities of detecting a transmission error, which may be missed when either of those checks is used alone.

An IS auditor needs to apply an assessment approach in validating reports depending on the situation (more evaluation is required when the organization has undergone a system change or when evaluating customized reports against standard reports of a widely used application). Methods to validate reports include the following:

- **Report distribution**—Output reports should be distributed according to authorized distribution parameters. Operations personnel should verify that reports are complete and delivered according to schedule. All reports should be logged prior to distribution. In most environments, processing output is spooled to a buffer or print spool on completion of job processing, where it waits for an available printer. Controls over access to the print spools are important to prevent reports from being deleted accidentally from print spools or directed to a different printer. In addition, changes to the output print priority can delay printing of critical jobs. Access to distributed reports can compromise confidentiality; therefore, physical distribution of reports should be controlled adequately. Reports containing sensitive data should be printed under secure, controlled conditions. Secure output drop-off points should be established. Output disposal should also be secured to ensure that no unauthorized access can occur. Electronic distribution should also be considered, and logical access controls should be put in place.
- **Balancing and reconciling**—Data processing application program output should be balanced routinely to the control totals. Audit trails should be provided to facilitate the tracking of transaction processing and the reconciliation of data.
- **Output error handling**—Procedures for reporting and controlling errors contained in the application program output should be established. The error report should be timely and delivered to the originating department for review and error correction.
- **Output report retention**—A record retention schedule should be adhered to firmly. Any governing legal regulations should be included in the retention policy.

- **Verification of receipt of reports**—To provide assurance that sensitive reports are properly distributed, the recipient should sign a log as evidence of receipt of output.

An IS auditor should be aware of existing concerns regarding record-retention policies for the organization and address legal requirements. Output can be restricted to particular IT resources or devices (e.g., a particular printer).

3.4.4 APPLICATION CONTROLS

Application controls are controls over the aforementioned input, processing and output functions. They include methods for ensuring that:

- Only complete, accurate and valid data are entered and updated in a computer system.
- Processing accomplishes the correct task.
- Processing results meet expectations.
- Data are maintained.

Application controls may consist of edit tests; totals; reconciliations; and identification and reporting of incorrect, missing or exception data. Automated controls should be coupled with manual procedures to ensure proper investigation of exceptions.

These controls help ensure data accuracy, completeness, validity, verifiability and consistency, thus achieving data integrity and data reliability. Implementation of these controls helps ensure that systems maintain integrity; applicable system functions operate as intended; and information contained by the system is relevant, reliable, secure and available when needed.

IS Auditor's Role in Reviewing Application Controls

When reviewing application controls, an IS auditor should perform the following activities:

- Identify the significant application components and the flow of transactions through the system and gain a detailed understanding of the application by reviewing the available documentation and interviewing appropriate

personnel.
- Identify the application control strengths and evaluate the impact of the control weaknesses.
- Develop a testing strategy.
- Test the controls to ensure their functionality and effectiveness by applying appropriate audit procedures.
- Evaluate the control environment by analyzing the test results and other audit evidence to determine that control objectives were achieved.
- Consider the operational aspects of the application to ensure its efficiency and effectiveness by comparing the system with efficient system design standards, analyzing procedures used and comparing them to management's objectives for the system.

When auditing application controls, an IS auditor should:
- Plan the audit, set audit objectives and identify risk associated with the application being audited.
- Identify the significant application components and the flow of information through the system and gain a detailed understanding of the application by reviewing the available documentation and interviewing appropriate personnel. Developing a DFD can help visualize the flow of information.

Flow of Information Through the System

A transaction flowchart provides information regarding key application processing controls. Points where transactions are entered, processed and posted should be reviewed for control weaknesses. The IS auditor should consider the following:
- Understand and evaluate interfaces, including APIs, in case the application connects with other applications.
- Identify the application key control strengths and evaluate the impact of the control weaknesses to develop a testing strategy by analyzing the accumulated information.
- Review application system documentation to provide an understanding of the functionality of the application. In many cases, it is not feasible to review the whole application documentation. Thus, a selective review should be performed. Any changes to applications should be documented properly. To gain an understanding of an application's development, an IS

auditor should review the following documentation:
  - **System development methodology documents**—Documents that describe the systems development approach used and include cost-benefit analysis and user requirements – **Functional design specifications**—Documents that provide a detailed explanation of the application. An understanding of key control points should be noted during review of the design specifications.
  - **Program changes**—Documents that detail any program change and provide evidence of authorization and a cross-reference to source code
  - **User manuals**—Documents that provide the foundation for understanding how the user is using the application. Often control weaknesses can be noted from the review of this document.
  - **Technical reference documentation**—Documents that include vendor-supplied technical manuals for purchased applications in addition to any in-house documentation. Access rules and logic usually are included in these documents.
- Test the effectiveness of application controls. The best practice for testing involves multiple levels of testing. See section 3.5, Testing Methodologies, for more information.
- Report the results. The report should include the audit objectives, tests conducted, test results, and findings and recommendations.

Risk Assessment Model to Analyze Application Controls

Conducting a risk assessment provides information relating to the inherent risk of an application. This risk assessment model can be based on many factors, which may include a combination of the following:

- The quality of internal controls
- Economic conditions
- Recent accounting system changes
- Time elapsed since last audit
- Complexity of operations
- Changes in operations/environment
- Recent changes in key positions
- Time in existence
- Competitive environment
- Assets at risk

- Prior audit results
- Staff turnover
- Transaction volume
- Regulatory agency impact
- Monetary volume
- Sensitivity of transactions
- Impact of application failure

3.4.5 USER PROCEDURES

User procedures that should be observed and tested by an IS auditor include:

- **SoD**—Ensures that no individual has the capability of performing more than one of the following processes: origination, authorization, verification or distribution. Observation and review of job descriptions and review of authorization levels and procedures may provide information regarding the existence and enforcement of SoD.
- **Authorization of input**—Provides evidence of input authorization via written authorization on input documents or with the use of unique passwords. This can be tested by looking through a sampling of input documents for proper authorization or reviewing computer access rules. Supervisor overrides of data validation and editing should be reviewed to ensure that automatic logging occurs. This override activity report should be tested for evidence of managerial review. Excessive overrides may indicate the need for modification of validation and editing routines to improve efficiency.
- **Balancing**—Verifies that run-to-run control totals and other application totals are reconciled on a timely basis. This may be tested by independent balancing or reviewing past reconciliations.
- **Error control and correction**—Provides evidence of appropriate review in the form of reports, research, timely correction and resubmission. Input errors and rejections should be reviewed prior to resubmission. Managerial review and authorization of corrections should be evidenced. Testing of this effort can be achieved by retabulating or reviewing past error corrections.
- **Distribution of reports**—Produces and maintains critical output reports in a secure area and distributes in an authorized manner. The distribution

process can be tested by observation and review of distribution output logs. Access to online output reports should be restricted. Online access may be tested through a review of the access rules or by monitoring user output.

- **Review and testing of access authorizations and capabilities**—Provides information regarding access levels (control tables). Access should be based on job descriptions and should provide for SoD. Testing can be performed through the review of access rules to ensure that access has been granted as management intended.
- **Activity reports**—Provide details, by user, of activity volume and hours. Activity reports should be reviewed to ensure that activity occurs only during authorized hours of operation.
- **Violation reports**—Record any unsuccessful and unauthorized access attempts. Violation reports should indicate the terminal location, date and time of attempted access. These reports should evidence managerial review. Repeated unauthorized access violations may indicate attempts to circumvent access controls. Testing may include review of follow-up activities.

3.4.6 DECISION SUPPORT SYSTEM

A decision support system (DSS) is an interactive system that provides the user with easy access to decision models and data from a wide range of sources in order to support semistructured decision-making tasks typically for business purposes. It is an informational application that is designed to assist an organization in making decisions through data provided by business intelligence tools (in contrast to an operational application that collects the data in the course of normal business operations).

Typical information that a decision support application might gather and present would be:

- Comparative sales figures between one week and the next
- Projected revenue figures based on new product sales assumptions
- The consequences of different decision alternatives given past experience in the described context

A DSS may present information graphically and may include an expert system or AI. Further, it may be aimed at business executives or some other

group of knowledge workers.

Characteristics of a DSS include:

- It is aimed at solving less structured, underspecified problems that senior managers face.
- It combines the use of models or analytic techniques with traditional data access and retrieval functions.
- It emphasizes flexibility and adaptability to accommodate changes in the environment and the decision-making approach of the users.

A principle of DSS design is to concentrate less on efficiency (i.e., performing tasks quickly and reducing costs) and more on effectiveness (i.e., performing the right task). Therefore, DSSs are often developed using 4GL tools that are less efficient but allow for flexible and easily modified systems. A DSS is often developed with a specific decision or well-defined class of decisions to solve; therefore, some commercial software packages that claim to be a DSS are nothing more than a DSS generator (tools with which to construct a DSS).

Design and Development

Prototyping is the most popular approach to DSS design and development. Prototyping usually bypasses the usual requirement definition. System requirements evolve through the user's learning process. The benefits of prototyping include the following:

- Learning is explicitly incorporated into the design process because of the iterative nature of the system design.
- Feedback from design iterations is rapid to maintain an effective learning process for the user.
- The user's expertise in the problem area helps the user suggest system improvements.
- The initial prototype must be inexpensive to create.

Implementation and Use

It is difficult to implement a DSS because of its discretionary nature. Using a DSS to solve a problem represents a change in behavior on the part of the user. Implementing a DSS is an exercise in changing an organization's

behavior. The main challenge is to get the users to accept the use of software. The following are the steps involved in changing behavior:

- **Unfreezing**—This step alters the forces acting on individuals such that the individuals are distracted sufficiently to change. Unfreezing is accomplished either through increasing the pressure for change or by reducing some of the threats of or resistance to change.
- **Moving**—This step presents a direction of change and the actual process of learning new attitudes.
- **Refreezing**—This step integrates the changed attitudes into the individual's personality.

Risk Factors

Developers should be prepared for eight implementation risk factors:

1. Nonexistent or unwilling users
2. Multiple users or implementers
3. Disappearing users, implementers or maintainers
4. Inability to specify purpose or usage patterns in advance
5. Inability to predict and cushion impact on all parties
6. Lack or loss of support
7. Lack of experience with similar systems
8. Technical problems and cost-effectiveness issues

Implementation Strategies

To plan for risk and prevent it from occurring:

- Divide the project into manageable pieces.
- Keep the solution simple.
- Develop a satisfactory support base.
- Meet user needs and institutionalize the system.

Assessment and Evaluation

The true test of a DSS lies in whether it improves a manager's decision-making, which is not easily measured. A DSS also rarely results in cost displacements such as a reduction in staff or other expenses. In addition, because a DSS is evolutionary in nature, it lacks neatly defined completion dates.

Using an incremental approach to DSS development reduces the need for evaluation. By developing one step at a time and achieving tangible results at the end of each step, the user does not need to make extensive commitments of time and money at the beginning of the development process.

The DSS designer and user should use broad evaluation criteria. These criteria should include:

- Traditional cost-benefit analysis
- Procedural changes, more alternatives examined and less time consumed in making the decision
- Evidence of improvement in decision-making
- Changes in the decision process

DSS Common Characteristics

DSSs share many common characteristics. These include:

- Oriented toward decision making
- Usually based on 4GL
- Surfable
- Linkable
- Drill-down
- Semaphores (signals to automatically alert when a decision needs to be made)
- Time series analysis
- What if (refers to scenario modeling, such as determining the end result of a changing variable or variables)
- Sensitivity analysis
- Goal-seeking
- Excellent graphic presentations
- Dynamic graphic, data editing
- Simulation

PART B: INFORMATION SYSTEMS IMPLEMENTATION

Information systems implementation is when the system is installed and moved into the production environment after appropriate system and users' acceptance testing. This is the stage at which:

- End users are notified.
- Data entry or conversions occur.
- Training takes place.
- Post-implementation reviews occur.

3.5 TESTING METHODOLOGIES

Integral to information systems implementation is the proper selection of testing methodologies, the development of testing plans fully traceable to requirements and the acquisition of essential resources to successfully complete testing. Once completed, testing provides confidence to stakeholders that a system or system component operates as intended and delivers the benefits realization as required at the start of project.

An IS auditor should understand the application of various forms of testing. An IS auditor should also understand how QA monitoring and evaluation contribute to the quality of an organization's internal processes (e.g., project management, software development process or IT service) and the quality of the final products produced by these processes (e.g., the system implemented or software developed).

Testing is an essential part of the systems development process that verifies and validates that a program, subsystem or application performs the functions for which it has been designed. Testing also determines whether the units being tested operate without any malfunction or adverse effect on other components of the system.

The variety of systems development methodologies and organizational requirements provide for a large range of testing schemes or levels. Each set of tests is performed with a different set of data and under the responsibility of different people or functions. An IS auditor plays a preventive or detective role in the testing process.

3.5.1 TESTING CLASSIFICATIONS

The following tests relate, to varying degrees, to the approaches that can be performed based on the size and complexity of the modified system:

- **Unit testing**—The testing of an individual program or module. Unit testing uses a set of test cases that focus on the control structure of the procedural design. These tests ensure that the internal operation of the program performs according to specification.
- **Interface or integration testing**—A hardware or software test that evaluates the connection of two or more components that pass information from one area to another. The objective is to take unit-tested modules and build an integrated structure dictated by design. The term "integration testing" is also used to refer to tests that verify and validate the functioning of the application under test with other systems, in which a set of data is transferred from one system to another.
- **System testing**—A series of tests designed to ensure that modified programs, objects, database schema, etc., which collectively constitute a new or modified system, function properly. These test procedures are often performed in a nonproduction test/development environment by software developers designated as a test team. The following specific analyses may be carried out during system testing:
  - Recovery testing—Checking the system's ability to recover after a software or hardware failure
  - Security testing—Making sure the modified/new system includes provisions for appropriate access controls and does not introduce any security holes that might compromise other systems
  - Load testing—Testing an application with large quantities of data to evaluate its performance during peak hours
  - Volume testing—Studying the impact on the application by testing with an incremental volume of records to determine the maximum volume of

records (data) the application can process
  - Stress testing—Studying the impact on the application by testing with an incremental number of concurrent users/services on the application to determine the maximum number of concurrent users/services the application can process
  - Performance testing—Comparing the system's performance to other equivalent systems using well-defined benchmarks
- **Final acceptance testing**—Performed after the system staff is satisfied with the system tests. Acceptance testing occurs during the implementation phase. During this testing phase, the defined methods of testing to apply should be incorporated into the organization's QA methodology. QA activities should proactively encourage that adequate levels of testing be performed on all software development projects. Final acceptance testing has two major parts: quality assurance testing (QAT), focusing on technical aspects of the application, and UAT, focusing on functional aspects of the application. QAT and UAT have different objectives and, therefore, should not be combined.

QAT focuses on the documented specifications and the technology employed. It verifies that the application works as documented by testing the logical design and the technology itself. It also ensures that the application meets the documented technical specifications and deliverables. QAT is performed primarily by the IT department. The participation of the end user is minimal and on request. QAT does not focus on functionality testing.

UAT supports the process of ensuring that the system is production-ready and satisfies all documented requirements. The methods include:
- Definition of test strategies and procedures
- Design of test cases and scenarios
- Execution of the tests
- Utilization of the results to verify system readiness

Acceptance criteria are defined elements that a deliverable must meet to satisfy the predefined needs of the user. A UAT plan must be documented for the final test of the completed system. The tests are written from a user's perspective and should use the system in a manner as close to production as

possible. For example, tests may be based around typical, predefined business process scenarios. If a new business process has been developed to accommodate the new or modified system, it should also be tested at this point. A key aspect of testing should also include testers seeking to verify that supporting processes integrate into the application in an acceptable fashion. Successful completion would generally enable a project team to hand over a complete integrated package of application and supporting procedures.

Ideally, UAT should be performed in a secure testing or staging environment. A secure testing environment, in which both source code and executable code are protected, helps to ensure that unauthorized or last-minute changes are not made to the system without going through the standard system maintenance process. The nature and extent of the tests will be dependent on the magnitude and complexity of the system change.

Even though acquired systems are tested by the vendor prior to distribution, these systems and any subsequent changes should be tested thoroughly by the end user and the system maintenance staff. These supplemental tests will help ensure that programs function as designed by the vendor and the changes do not interact adversely with existing systems. In the case of acquired software, after attending to the changes during testing by the vendor, the accepted version should be controlled and used for implementation. In the absence of controls, the risk of introducing malicious patches/Trojan horse programs is very high.

Some organizations rely on integrated test facilities (ITFs). Test data usually are processed in production-like systems. This confirms the behavior of the new application or modules in real-life conditions. These conditions include peak volume and other resource-related constraints. In this environment, IS will perform tests with a set of fictitious data whereas client representatives use extracts of production data to cover the most possible scenarios as well as some fictitious data for scenarios that would not be tested by the production data. In some organizations that use a subset of production data in a test environment, such production data may be altered to scramble the data so that the confidential nature of the data is obscured from the tester. This is often the case when the acceptance testing is done by team members who, under

usual circumstances, would not have access to such production data.

Once acceptance testing is complete, a certification and accreditation process is performed after the system is implemented and in operation for some time to produce the evidence needed for these processes. This process includes evaluating program documentation and testing effectiveness and results in a final decision for deploying the business application system. For information security issues, the evaluation process includes reviewing security plans, the risk assessments performed and test plans, and the evaluation process results in an assessment of the effectiveness of the security controls and processes to be deployed. Generally involving security staff and the business owner of the application, this process provides some degree of accountability to the business owner regarding the state of the system that he/she will accept for deployment.

When the tests are completed, an IS auditor should issue an opinion to management as to whether the system meets the business requirements, has implemented appropriate controls, and is ready to be migrated to production. This report should specify the deficiencies in the system that need to be corrected and should identify and explain the risk that the organization is taking by implementing the new system.

Other Types of Testing

Other types of testing include the following:

- **Alpha and beta testing**—The two phases of testing software undergoes before being considered finished. The first stage, called alpha testing, is often performed only by users within the organization developing the software (i.e., systems testing). The second stage, called beta testing, a form of UAT, generally involves a limited number of external users. This involves real-world exposure, sending the beta version of the product to independent test sites or offering it free to interested users.
- **Pilot testing**—A preliminary test that focuses on specific and predetermined aspects of a system. It is not meant to replace other testing methods, but rather to provide a limited evaluation of the system. POCs are early pilot tests—usually over interim platforms and with only basic functionalities.

- **White box testing**—A test that assesses the effectiveness of software program logic. Specifically, test data are used in determining procedural accuracy or conditions of a program's specific logic paths (i.e., applicable to unit and integration testing). However, testing all possible logic paths in large information systems is not feasible and would be cost-prohibitive; therefore, white box testing is used on a select basis only.
- **Black box testing**—An integrity-based form of testing associated with testing components of an information system's "functional" operating effectiveness without regard to any specific internal program structure. It is applicable to integration (interface) and UAT processes.
- **Function/validation testing**—Similar to system testing but often used to test the functionality of the system against the detailed requirements to ensure that the software that has been built is traceable to customer requirements (i.e., Are we building the right product?)
- **Regression testing**—The process of rerunning a portion of a test scenario or test plan to ensure that changes or corrections have not introduced new errors. The data used in regression testing should be the same as the data used in the original.
- **Parallel testing**—The process of feeding test data into two systems—the modified system and an alternative system (possibly the original system)—and comparing the results. The purpose of parallel testing is to determine whether the new application performs in the same way as the original system and meets end-user requirements.
- **Sociability testing**—Tests to confirm that the new or modified system can operate in its target environment without adversely impacting existing systems. This should cover the platform that will perform primary application processing and interfaces with other systems and, in a client server or web development, those that perform changes to the desktop environment.

3.5.2 SOFTWARE TESTING

Test plans identify the specific portions of the system to be tested and may include a categorization of types of deficiencies that can be found during the test. Categories of such deficiencies may be system defects, incomplete requirements, designs, specifications, or errors in the test case itself. Test

plans also specify severity levels of problems found, as well as guidelines on identifying the business priority. The tester determines the severity of the problem found during testing. Based on the severity level, the problem may be fixed prior to implementation or may be noted for correction following implementation. The project sponsor, end-user management and the project manager decide early in the test phase on the severity definitions.

Test plans also identify test approaches, such as the following two reciprocal approaches, to software testing:

- **Bottom up**—Testing begins with atomic units, such as programs or modules, and works upward until a complete system testing has taken place. The advantages follow:
  - There is no need for stubs or drivers.
  - Testing can be started before all programs are complete.
  - Errors in critical modules are found early.
- **Top down**—Testing follows the opposite path, either in depth-first or breadth-first search order. The advantages follow:
  - Tests of major functions and processing are conducted early.
  - Interface errors can be detected sooner.
  - Confidence in the system is increased because programmers and users actually see a working system.

Generally, most application testing of large systems follows a bottom-up testing approach that involves ascending levels of integration and testing (e.g., unit or program, subsystem/integration, system):

- **Conduct and report test results**—Describe resources implied in testing, including personnel involved and information resources/facilities used during the test as well as actual versus expected test results. Results reported, along with the test plan, should be retained as part of the system's permanent documentation.
- **Address outstanding issues**—Identify errors and irregularities from the actual tests conducted. When such problems occur, the specific tests in question have to be redesigned in the test plan until acceptable conditions occur when the tests are redone.

3.5.3 DATA INTEGRITY TESTING

Data integrity testing is a set of substantive tests that examines accuracy, completeness, consistency and authorization of data presently held in a system. It employs testing similar to that used for input control. Data integrity tests indicate failures in input or processing controls. Controls for ensuring the integrity of accumulated data in a file can be exercised by regularly checking data in the file. When this checking is done against authorized source documentation, it is common to check only a portion of the file at a time. Because the whole file is regularly checked in cycles, the control technique is often referred to as cyclical checking.

Two common types of data integrity tests are relational and referential integrity tests:

- **Relational integrity tests**—Performed at the data element and record-based levels. Relational integrity is enforced through data validation routines built into the application or by defining the input condition constraints and data characteristics at the table definition in the database stage. Sometimes it is a combination of both.
- **Referential integrity tests**—Define existence relationships between entities in different tables of a database that needs to be maintained by the DBMS. It is required for maintaining interrelation integrity in the relational data model. Whenever two or more relations are related through referential constraints (primary and foreign key), it is necessary that references be kept consistent in the event of insertions, deletions and updates to these relations. Database software generally provides various built-in automated procedures for checking and ensuring referential integrity. Referential integrity checks involve ensuring that all references to a primary key from another table (i.e., a foreign key) actually exist in their original table. In nonpointer databases (e.g., relational), referential integrity checks involve making sure that all foreign keys exist in their original table.

Data Integrity in Online Transaction Processing Systems In multiuser transaction systems, it is necessary to manage parallel user access to stored data typically controlled by a DBMS and deliver fault tolerance. Of particular importance are four online data integrity requirements known collectively as the ACID principle:

- **Atomicity**—From a user perspective, a transaction is either completed in

its entirety (i.e., all relevant database tables are updated) or not at all. If an error or interruption occurs, all changes made up to that point are backed out.
- **Consistency**—All integrity conditions in the database are maintained with each transaction, taking the database from one consistent state into another consistent state.
- **Isolation**—Each transaction is isolated from other transactions, so each transaction accesses only data that are part of a consistent database state.
- **Durability**—If a transaction has been reported back to a user as complete, the resulting changes to the database survive subsequent hardware or software failures.

3.5.4 APPLICATION SYSTEMS TESTING

Testing the effectiveness of application controls involves analyzing computer application programs, testing computer application program controls, or selecting and monitoring data process transactions. Testing controls by applying appropriate audit procedures is important to ensure their functionality and effectiveness. Methods and techniques for each category are described in **figure 3.20**.

To facilitate the evaluation of application system tests, an IS auditor may also want to use generalized audit software. This is particularly useful when specific application control weaknesses are discovered that affect, for example, updates to master file records and certain error conditions on specific transaction records. Additionally, generalized audit software (GAS) can be used to perform certain application control tests, such as parallel simulation, in comparing expected outcomes to live data.

Figure 3.20—Testing Application Systems			
Analyzing Computer Application Programs			
Technique	**Description**	**Advantages**	**Disadvantages**
Snapshot	• Records flow of designated transactions through logic paths within programs	• Verifies program logic	• Requires extensive knowledge of the IS environment
Mapping	• Identifies specific program logic that has not been tested and analyzes programs during execution to indicate whether program statements have been executed	• Increases efficiency by identifying unused code • Identifies potential exposures	• Cost of software
Tracing and tagging	• Tracing shows the trail of instructions executed during an application. • Tagging involves placing an indicator on selected transactions at input and using tracing to track them.	• Provides an exact picture of sequence of events, and is effective with live and simulated transactions	• Requires extensive amounts of computer time, an intimate knowledge of the application program and additional programming to execute trace routines
Test data/deck	• Simulates transactions through real programs	• May use actual master files or dummies • Source code review unnecessary • Can be used on a surprise basis • Provides objective review and verification of program controls and edits • Initial use can be limited to specific program functions minimizing scope and complexity. • Requires minimal knowledge of the IS environment	• Difficult to ensure that the proper program is checked • Risk of not including all transaction scenarios • Requires good knowledge of application systems • Does not test master file and master file records
Base-case system evaluation	• Uses test data sets developed as part of a comprehensive testing of programs • Verifies correct system operations before acceptance, as well as periodic revalidation	• Comprehensive testing verification and compliance testing	• Extensive effort to maintain data sets • Close cooperation is required among all parties.
Parallel operation	• Processes actual production data through existing and newly developed programs at the same time and compares results and is used to verify changed production prior to replacing existing procedures	• Verifies new system before discontinuing the old one	• Added processing costs
Integrated testing facility	• Creates a fictitious file in the database with test transactions processed simultaneously with live data	• Periodic testing does not require separate test process.	• Need for careful planning • Need to isolate test data from production data
Parallel simulation	• Processes production data using computer programs that simulate application program logic	• Eliminates need to prepare test data	• Programs must be developed.
Transaction selection programs	• Use audit software to screen and select transactions input to the regular production cycle	• Independent of production system • Controlled by the auditor • Requires no modification to production systems	• Cost of development and maintenance

Embedded audit data collection	• Software embedded in host computer applications screens. It selects input transactions and generates transactions during production. Usually, it is developed as part of system development. Types include: – Systems control audit review file (SCARF): Auditor determines reasonableness of tests incorporated into normal processing. It provides information for further review. – Sample audit review file (SARF): Randomly selects transactions to provide representative file for analysis	• Provides sampling and productions statistics	• High cost of development and maintenance • Auditor independence issues
Extended records	• Gathers all data that have been affected by a particular program	• Records are put into one convenient file.	• Adds to data storage costs and overhead, and to system development costs

Automated Application Testing

Test data generators can be used to systematically generate random data that can be used to test programs. The generators work by using the field characteristics, layout and values of the data. In addition to test data generators, there are interactive debugging aids and code logic analyzers available to assist in the testing activities.

3.5.5 IS AUDITOR'S ROLE IN INFORMATION SYSTEMS TESTING

Testing is crucial in determining that user requirements have been validated, the system is performing as anticipated and internal controls work as intended. Therefore, it is essential that an IS auditor be involved in reviewing this phase and perform the following:

- Review the test plan for completeness; indicate evidence of user participation, such as user development of test scenarios and/or user sign-off of results; and consider rerunning critical tests.
- Reconcile control totals and converted data.
- Review error reports for their precision in recognizing erroneous data and resolution of errors.
- Verify cyclical processing for correctness (month-end, year-end processing, etc.).
- Verify accuracy of critical reports and output used by management and other stakeholders.

- Interview end users of the system for their understanding of new methods, procedures and operating instructions.
- Review system and end-user documentation to determine its completeness and verify its accuracy during the test phase.
- Review parallel testing results for accuracy.
- Verify that system security is functioning as designed by developing and executing access tests.
- Review unit and system test plans to determine whether tests for internal controls are planned and performed.
- Review the UAT and ensure that the accepted software has been delivered to the implementation team. The vendor should not be able to replace this version.
- Review procedures used for recording and following through on error reports.

3.6 CONFIGURATION AND RELEASE MANAGEMENT

The effective and efficient development and maintenance of complicated IT systems requires that rigorous configuration, change and release management processes be implemented and adhered to within an organization. These processes provide systematic, consistent and unambiguous control on attributes of IT components comprising the system (hardware, software, firmware, and network connectivity including physical connecting media wire, fiber and radio frequency [RF]). Knowledge of the configuration status of computing environments is critical to system reliability, availability and security along with achieving timely maintenance of these systems. Changes to IT systems must be carefully assessed, planned, tested, approved, documented and communicated to minimize any undesirable consequences to the business processes.

An IS auditor should be aware of the tools available for managing configuration, change and release management and of the controls in place to ensure SoD between development staff and the production environment.

Because of the difficulties associated with exercising control over both system and programming maintenance activities, more organizations

implement configuration management systems. In many cases, regulatory requirements mandate these levels of control to provide a high degree of reliability and repeatability in all associated system processes. In a configuration management system, maintenance requests must be formally documented and approved by a change control group (e.g., configuration control boards). In addition, careful control is exercised over each stage of the maintenance process via checkpoints, reviews and sign-off procedures.

Configuration management involves procedures throughout the system hardware and software life cycle (from requirements analysis to maintenance) to identify, define and baseline software items in the system and thus provide a basis for problem management, change management and release management.

The process of checking out prevents or manages simultaneous code edits, with hardware, network and system architects reviewing and approving the changes or updates to both the hardware asset and inventory tracking systems.

Checking in is the process of moving an item to the controlled environment. When a change is required (and supported by a change control form), the configuration manager will check out the item. Once the change is made, it can be checked using a different version number. The process of checking out also prevents or manages simultaneous code edits. With hardware, network and system architects review and approve the changes or updates to both the hardware asset and the inventory tracking systems.

For configuration management to work, management support is critical. The configuration management process is implemented by developing and following a configuration management plan and operating procedures. This plan should not be limited to just the software developed but should also include all system documentation, test plans and procedures.

Commercial software products are often used to automate some processes. Such tools should allow control to be maintained for applications software from the outset of system analysis and design to running live. Configuration

management tools will support change management and release management through the:

1. Identification of items affected by a proposed change to assist with impact assessment (functional, operational and security)
2. Recording configuration items affected by authorized changes 3. Implementation of changes in accordance with authorization records
4. Registering of configuration item changes when authorized changes and releases are implemented
5. Recording of baselines that are related to releases (with known consequences) to which an organization would revert if an implemented change fails
6. Preparing a release to avoid human errors and resource costs

A new version of the system (or builds) should be built only from the baselined items. The baseline becomes the trusted recovery source for these systems and applications.

From an IS audit perspective, effective use of configuration management software provides important evidence of management's commitment to careful control over the maintenance process.

3.7 SYSTEM MIGRATION, INFRASTRUCTURE DEPLOYMENT AND DATA CONVERSION

New software applications tend to be more comprehensive and integrated than older applications. Furthermore, organizations rely increasingly on data warehouses, models and simulation for decision-making; thus, importing data from old (and legacy) systems into the new application is crucial. Data format, coding, structure and integrity are to be preserved or properly translated. A migration scenario must be set up and a rollback plan needs to be in place. There are many direct (old to new application) and indirect (using interim repositories) strategies and tools. Data conversion is a one-time task in many development projects. The importance of correct results is critical, and success depends on the use of good practices by the development team as the programmed input checks under development will not be available for the conversion. Source data must be correctly characterized and the destination

database must accommodate all existing data values. Resulting data should be carefully tested. Steps for the conversion that are developed in the test environment must be recorded so they can be repeated on the production system.

An IS auditor should ensure that any tools and techniques selected for the process are adequate and appropriate, data conversion achieves the necessary objectives without data loss or corruption, and any loss of data is both minimal and formally accepted by user management.

3.7.1 DATA MIGRATION

A data conversion (also known as data porting) is required if the source and target systems use different field formats or sizes, file/database structures, or coding schemes. For example, a number may be stored as text, floating point or binary-coded-decimal.

Conversions are often necessary when the source and target systems are on different hardware and/or OS platforms, and different file or database structures (e.g., relational database, flat files, virtual storage access method) are used.

The objective of data conversion is to convert existing data into the new required format, coding and structure while preserving the meaning and integrity of the data. The data conversion process must provide some means, such as audit trails and logs, to allow for the verification of the accuracy and completeness of the converted data. This verification of accuracy and completeness may be performed through a combination of manual processes, system utilities, vendor tools and one-time-use special applications.

A large-scale data conversion can potentially become a project within a project as considerable analysis, design and planning will be required. Among the steps necessary for a successful data conversion are:

- Determining what data should be converted using programs and what, if any, should be converted manually
- Performing any necessary data cleansing ahead of conversion
- Identifying the methods to be used to verify the conversion, such as

automated file comparisons, comparing record counts and control totals, accounting balances, and individual data items on a sample basis
- Establishing the parameters for a successful conversion (e.g., Is 100 percent consistency between the old and new systems necessary, or will some differences within defined ranges be acceptable?)
- Scheduling the sequence of conversion tasks
- Designing audit trail reports to document the conversion, including data mappings and transformations
- Designing exception reports to record any items that cannot be converted automatically
- Establishing responsibility for verifying and signing off on individual conversion steps and accepting the overall conversion
- Developing and testing conversion programs, including functionality and performance
- Performing one or more conversion dress rehearsals to familiarize persons with the sequence of events and their roles, and testing the conversion process end to end with real data
- Controlling the outsourcing of the conversion process with a proper agreement covering nondisclosure, data privacy, data destruction and other warranties
- Running the actual conversion with all necessary personnel onsite or able to be contacted

A successful data migration delivers the new system on time, on budget and with the required quality. The data migration project should be carefully planned and use appropriate methodologies and tools to minimize the risk of:
- Disruption of routine operations
- Violation of the security and confidentiality of data
- Conflicts and contention between legacy and migrated operations
- Data inconsistencies and loss of data integrity during the migration process

The data model and the new application model should be stored in an enterprise repository. Using a repository allows a simulation of the migration scenario and traceability during the project. An enterprise repository enables an overview of the reengineering and data migration process (e.g., which modules and entities are in which stage, such as in service or already

migrated). These models will be modified in the course of the processes described in the following sections.

Refining the Migration Scenario

In order to determine the scope of the implementation project, module analysis should be undertaken to identify the affected functional modules and data entities. The plan of the implementation project should be refined based on this information and an analysis of business requirements.

The next step is to develop a migration plan. This is a detailed listing of tasks for the production deployment of a new system. Within this plan decision points are defined to make “go” or “nogo” decisions. The following processes require decision points:

- **Support migration process**—A support process to administer the enterprise repository must be implemented. Because this repository should be used on completion of the project to manage the software components of the new architecture, this process should be capable of supporting future development processes. The enterprise repository administration and report generation support the migration by supporting the reverse engineering of changes in the legacy architecture and facilitating the creation of impact analysis reports.
- **Migration infrastructure**—The project develops specifications for the infrastructure of the migration project. This approach ensures consistency and increases confidence in the functionality of the fallback scenario. The migration project team completes a high-level analysis of the legacy and new data models to establish links between them that will be refined later. The migration infrastructure is the basis for specifying the following components:
  - Data redirector (temporary adapters)—Good practices suggest the staged deployment of applications to minimize the end-user impact of their implementation and limit the risk by having a fallback scenario with minimum impact. For this reason, an infrastructure component is needed to handle distributed data on different platforms within distributed applications. The design of a data redirector on the new architecture corresponds to service-oriented architectures and should cover features such as access to the not-yet-migrated legacy data during run time, data

consistency due to the usage of standards such as X/Open XA interface, and a homogeneous new architecture.

– Data conversion components—The need to create an enterprise data model to eliminate data redundancies and inconsistencies often is identified. For this reason, infrastructure components to transform the legacy data model to the new data model must be provided. These components can be described as follows:
  - Unload components to copy the data (either as is or suitably modified to align with the data format of target system) in legacy databases that have been identified for migration
  - Transfer components to execute the data transfer from the legacy system to the new system
  - Load components to execute the load of the data into the new database

Software packages that support data migration, such as ERP and document management software, should be acquired as soon as the software evaluation is done. The data conversion plan should be based on the available databases and migration tools provided by the selected vendor(s).

The decision on which method to use for data conversion has to be made as part of the implementation project and should be based transaction volume and change degree of the data model.

Fallback (Rollback) Scenario

Not all new system deployments go as planned. To mitigate the risk of downtime for mission-critical systems, good practices dictate that the tools and applications required to reverse the migration are available prior to attempting the production cutover. Some or all of these tools and applications may need to be developed as part of the project.

Components have to be delivered that can back out all changes and restore data to the original applications in the case of nonfunctioning new applications. Two types of components should be considered as part of a fallback contingency plan.

The first consists of:

- Unload components to execute the unloading of the data from the new data structures
- Transfer components for the data conversion
- Load components to execute the loading of the data into the legacy data structures

The second consists of:
- A log component to log the data modifications within the new data model during runtime within the service layer
- Transfer components for the data conversion
- Load components to execute the load of the data into the legacy data structures

Another important consideration is the new system's data structure. This can be determined by reading the software user guides, analyzing the entity relationship diagrams, understanding the relationships between data elements and reviewing definitions of key terms (such as "entity" and "record") in the new system.

Next, it is important to review the decisions on how business processes should be conducted in the new system. Changes are identified, and the output of this exercise is a table of new data terminology against current definitions of data elements. In this step, the project team identifies how current data are defined in the new system. Following this step, a data cleanup is completed to eliminate inconsistencies in the current database, if possible, and duplications of data sets are discovered and resolved. The rules of conversion are defined and documented, with the objective of ensuring the business processes executed in the new system yield results that maintain data integrity and relationships.

Data conversion rules are programmed by the software development team. Data conversion scripts are created to convert the data from the old database to the new database. These are tested on a discrete selection of data that is carefully selected to include all cases. This process is referred to as "program" or "unit testing." Following the sign-off of data conversion scripts by programmers, the scripts are run on a test copy of the production database.

The values of data are verified by executing assessments including business process tests. Users and developers complete cycles of testing until conversion scripts are fine-tuned. After testing has been completed, the next step is to promote the converted database to production.

The key points to be taken into consideration in a data conversion project are to ensure:

- Completeness of data conversion
- Integrity of data
- Storage and security of data under conversion
- Consistency of data
- Continuity of data access

The last copy of the data before conversion from the old platform and the first copy of the data after conversion to the new platform should be maintained separately in the archive for any future reference.

3.7.2 CHANGEOVER (GO-LIVE OR CUTOVER) TECHNIQUES

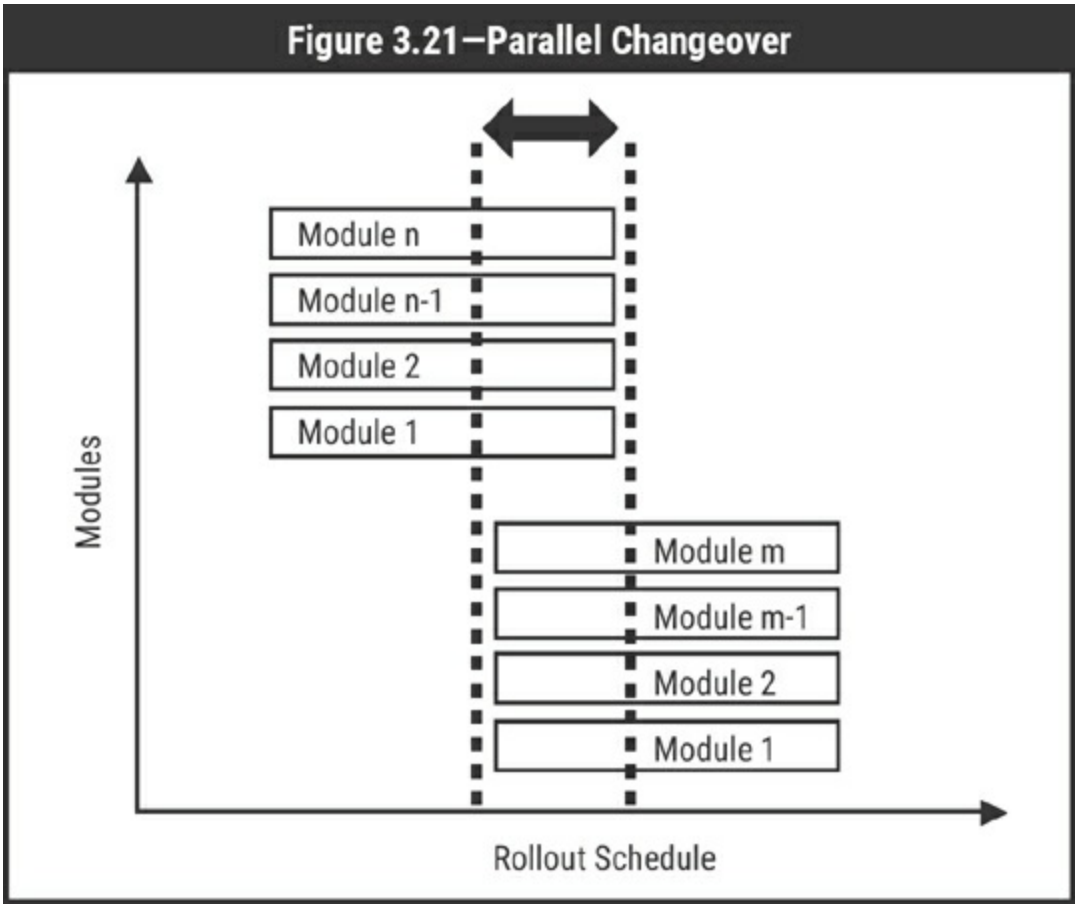
Changeover refers to an approach to shift users from using the application from the existing (old) system to the replacing (new) system. This is appropriate only after testing the new system with respect to its program and relevant data. This is sometimes called the “go-live” technique because it enables the start of the new system. This approach is also called the “cutover” technique because it helps in cutting out from the older system and moving over to the newer system.

This technique can be achieved in three different ways.

Parallel Changeover

This technique includes running the old system, then running both the old and new systems in parallel, and, finally, fully changing over to the new system after gaining confidence in the working of the new system. With this approach, the users will have to use both systems during the period of overlap. This will minimize the risk of using the newer system and, at the same time, help in identifying problems, issues or any concerns that the user

comes across in the newer system in the beginning. After a period of overlap, the user gains confidence and assurance in relying on the newer system. At this point, the use of the older system is discontinued, and the new system becomes totally operational. Note in **figure 3.21** that the number (m, n, respectively) of modules in the new and old systems may be different.



Phased Changeover

In this approach the older system is broken into deliverable modules. Initially, the first module of the older system is phased out using the first module of the newer system. Then, the second module of the older system is phased out, using the second module of the newer system, and so forth until reaching the last module. Thus, the changeover from the older system to the newer system takes place in a preplanned, phased manner. See **figure 3.22**.

Some of the risk that may exist in the phased changeover includes:

- Resource challenges (both on the IT side—to be able to maintain two

unique environments such as hardware, OSs, databases and code; and on the operations side—to be able to maintain user guides, procedures and policies, definitions of system terms, etc.)
- Extension of the project life cycle to cover two systems
- Change management for requirements and customizations to maintain ongoing support of the older system

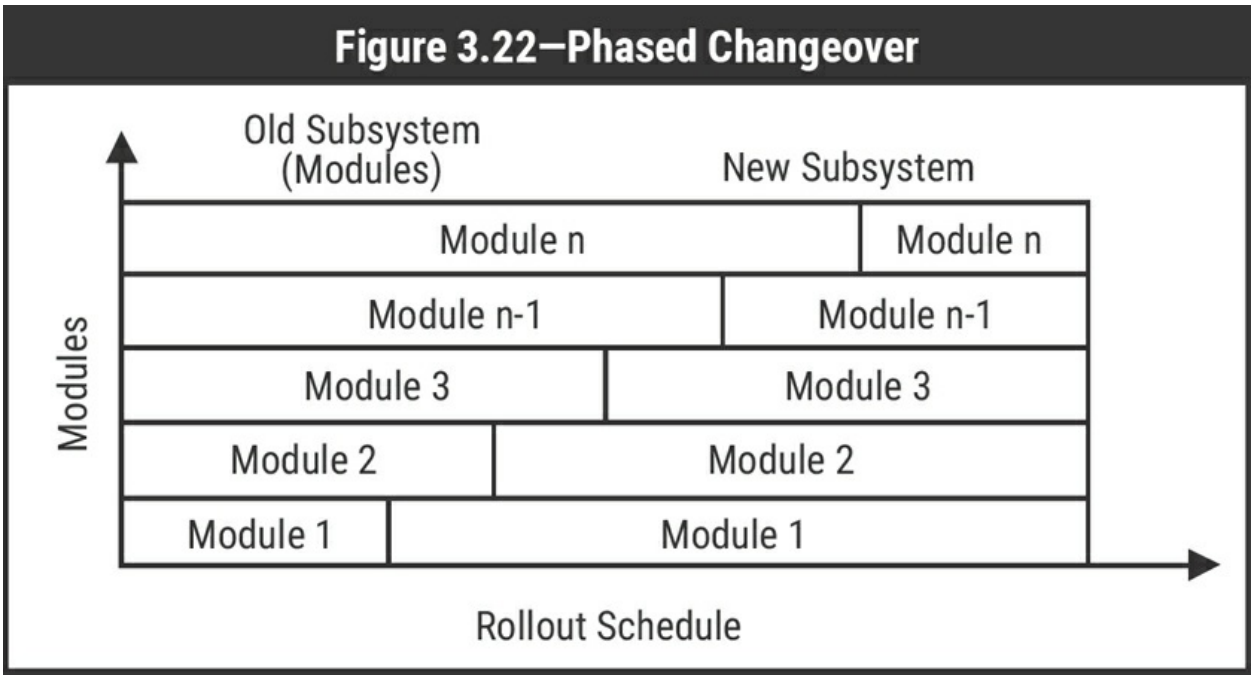
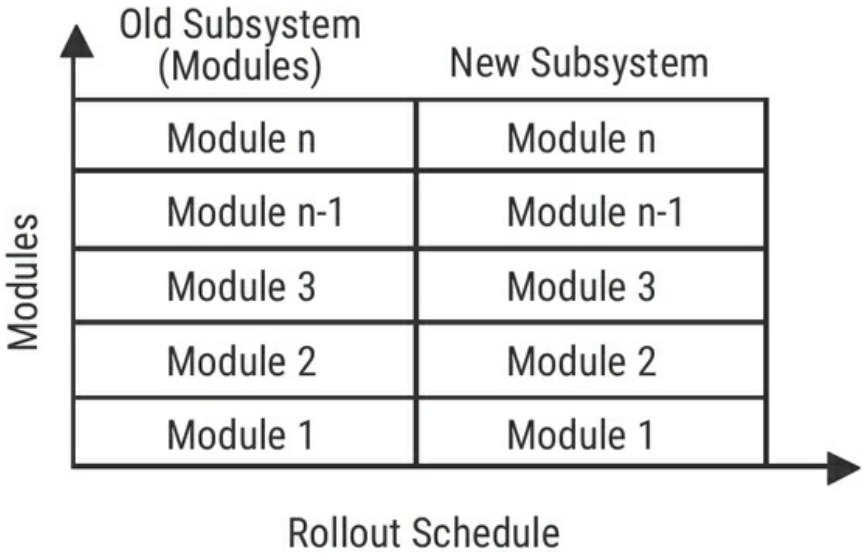


Figure 3.22—Phased Changeover

Abrupt Changeover

In this approach the newer system is changed over from the older system on a cutoff date and time, and the older system is discontinued once changeover to the new system takes place. See **figure 3.23**.

Figure 3.23—Abrupt Changeover



Changeover to the newer system involves four major steps or activities:

1. Conversion of files and programs; test running on test bed
2. Installation of new hardware, OS, application system and the migrated data
3. Training employees or users in groups
4. Scheduling operations and test running for go-live or changeover

Some of the risk items related to changeover include:

- Asset safeguarding
- Data integrity
- System effectiveness
- System efficiency
- Change management challenges (depending on the configuration items considered)
- Duplicate or missing records (possible existence of duplicate or erroneous records if data cleansing is not done correctly)

3.7.3 SYSTEM IMPLEMENTATION

Implementation is initiated only after a successful testing phase. The system should be installed according to the organization’s change control

procedures.

An IS auditor should verify that appropriate sign-offs have been obtained prior to implementation and perform the following:

- Review the programmed procedures used for scheduling and running the system along with system parameters used in executing the production schedule.
- Review all system documentation to ensure its completeness and confirm that all recent updates from the testing phase have been incorporated.
- Verify all data conversions to ensure that they are correct and complete before implementing the system in production.

Implementation Planning

Once developed and ready for operation, the new system delivered by the project will need an efficient support structure. It is not enough to set up roles for a support structure and name people to fulfill these roles. Support personnel will need to acquire new skills. Workload has to be distributed, in order for the right people to support the right issues; thus, new processes have to be developed while respecting the specificities of IT department requirements. Additionally, an infrastructure dedicated for support staff has to be made available. For these and other reasons, setting up a support structure normally is a project in itself and requires planning, a methodology and good practices adaptation from past experiences.

The objective of such a project is to develop and establish the to-be support structure for the new technical infrastructure. The main goals are to accomplish the following:

- Provide appropriate support structures for first-, second- and third-line support teams.
- Provide a single point of contact.
- Provide roles and skills definitions with applicable training plans.

Often the project sponsor’s organization operates and supports a legacy solution and will implement a new system environment based on new system architecture. The existing support procedures and the organizational units will have to maintain the future system to provide the appropriate level of support

for the new platform as well as for the old one.

To achieve significant success in updating staff on changes to the business process and introducing new software, it is necessary to address some important questions such as:

- How can the existing support staff be involved in the setup of the new project without neglecting the currently running system?
- What is the gap of knowledge/skills that must be addressed in the training plan?
- How large is the difference from the current legacy environment operation to the operation of the new platform?

Generally, a transition project should conform to the following guidelines:

- There should be a smooth transition from the existing platform to the new platform, without any negative effect on users of the system.
- There should be maximum employment of the existing support staff to operate the new system environment and keep the effort of new hires at a minimum level.

A primary challenge is to manage the phases from build, to integrate, to migrate, and for the phasing-out of the existing system and the phasing-in of the new one. The migration cannot be accomplished via a single event. Instead, a step-by-step transition of the affected services must take place. Further, the implemented processes for a legacy environment might be different from what may be implemented with the new platform and any changes must be communicated to users and system support staff.

Implementation Plan/Knowledge Transfer Plan

In accordance with good practices, the transfer should follow the shadowing and relay-baton method. Shadowing gives staff the opportunity to become accustomed to the system by observation. The relay-baton approach is the best suitable concept to transfer knowledge and also to transfer responsibility in a transparent way. The metaphor of the relay-baton expresses exactly what must be achieved—that is, knowledge is transferred in small portions.

Training Plan

After the roles and responsibilities are defined, they will be documented in the form of a chart to allow for a clear and easy-to-read overview.

For example, a staff training plan should show all of the required training in terms of:

- Content
- Scheduling information
- Duration
- Delivery mechanism (classroom and/or web-based)
- Train-the-trainer concept

The plan should consider the role definitions and skill profiles for the new to-be structure, and the results of the gap analysis. The plan considers that the staff who need to be trained must still run the current system, so a detailed coordination with the daily business tasks is maintained.

The following list gives an example of work tasks defined to fulfill the overall project goal:

- Collate existing support structure documentation.
- Review the existing IT organization model.
- Define the new support organization structure.
- Define the new support processes.
- Map the new process to the organization model.
- Execute the new organization model.
- Establish support functions.
- Develop communications material for support staff.
- Conduct briefing and training sessions.
- Review mobilization progress.
- Transfer to the new organization structure.
- Review the preceding items.

3.7.4 SYSTEM CHANGE PROCEDURES AND THE PROGRAM MIGRATION PROCESS

Following implementation and stabilization, a system enters into the ongoing development or maintenance stage. This phase continues until the system is retired. The phase involves those activities required to either correct errors in

the system or enhance the capabilities of the system.

In this regard, an IS auditor should consider the following:

- The existence and use of a methodology for authorizing, prioritizing and tracking system change requests from the user
- Whether emergency change procedures are addressed in the operations manuals
- Whether change control is a formal procedure for the user and the development groups
- Whether the change control log ensures all changes shown were resolved
- The user's satisfaction with the turnaround—timeliness and cost—of change requests
- The adequacy of the security access restrictions over production source and executable modules
- The adequacy of the organization's procedures for dealing with emergency program changes
- The adequacy of the security access restrictions over the use of the emergency logon IDs

For a selection of changes on the change control log:

- Determine whether changes to requirements resulted in appropriate change-development documents, such as program and operations documents.
- Determine whether changes were made as documented
- Determine whether current documentation reflects the changed environment.
- Evaluate the adequacy of the procedures in place for testing system changes.
- Review evidence (test plans and test results) to ensure that procedures are carried out as prescribed by organizational standards.
- Review the procedures established for ensuring executable and source code integrity.
- Review production executable modules and verify there is one and only one corresponding version of the program source code.
- Check the technical controls of the change management tool.

Additionally, an IS auditor should review the overall change management

process for possible improvements in acknowledgment, response time, response effectiveness and user satisfaction with the process.

Critical Success Factors

Critical success factors of planning the implementation include the following:

- To avoid delays, the appropriate skilled staff must attend workshops and participate for the entire project duration.
- The documentation needed for carrying out the work needs to be ready at project initiation.
- Decision-makers must be involved at all steps to ensure all necessary decisions can be made.

End-user Training

The goal of a training plan is to ensure that the end user can become self-sufficient in the operation of the system. One of the most important keys in end-user training is to ensure that training is considered and a training project plan is created early in the development process. A strategy can be developed that would take into consideration the timing, extent and delivery mechanisms.

The training should be piloted using a cross-section of users to determine how best to customize the training to the different user groups. Following the pilot, the training approach can be adjusted as necessary, based on the feedback received from the pilot group.

Separate classes should be developed for individuals who will assist in the training process. These train-the-trainer classes also provide useful feedback for improving the content of the training program.

The timing of the delivery of training is very important. If training is delivered too early, users will forget much of the training by the time the system actually goes into production. If training is delivered too late, there will not be enough time to obtain feedback from the pilot group and implement the necessary changes into the main training program. Training classes should be customized to address skill level and needs of users based on their role within the organization.

To develop the training strategy, an organization must name a training administrator. The training administrator will identify users who need to be trained with respect to their specific job functions. Consideration should be given to the following format and delivery mechanisms:

- Case studies
- Role-based training
- Lecture and breakout sessions
- Modules at different experience levels
- Practical sessions on how to use the system
- Remedial computer training (if needed)
- Online sessions on the web or on a CD-ROM

It is important to have a library of cases or tests, including user errors and the system response to those errors. The training administrator should record student information in a database or spreadsheet, including student feedback for improving the training course.

3.7.5 SYSTEM SOFTWARE IMPLEMENTATION

System software implementation involves identifying features, configuration options and controls for standard configurations to apply across the organization. Additionally, implementation involves testing the software in a nonproduction environment and obtaining some form of certification and accreditation to place the approved OS software into production.

3.7.6 CERTIFICATION/ACCREDITATION

Certification is a process by which an assessor organization performs a comprehensive assessment against a standard of management and operational and technical controls in an information system. The assessor examines the level of compliance in meeting certain requirements such as standards, policies, processes, procedures, work instructions and guidelines—requirements made in support of accreditation. The goal is to determine the extent to which controls are implemented correctly, operating as intended and producing the desired outcome with respect to meeting the system’s security requirements. The results of a certification are used to reassess the risk and update the system security plan, thus providing the factual basis for an

authorizing official to render an accreditation decision.

Accreditation is the official management decision (given by a senior official) to authorize operation of an information system and to explicitly accept the risk to the organization's operations, assets or individuals based on the implementation of an agreed-upon set of requirements and security controls. Security accreditation provides a form of QC and challenges managers and technical staff at all levels to implement the most effective security controls possible in an information system, given mission requirements, and technical, operational and cost/schedule constraints.

By accrediting an information system, a senior official accepts responsibility for the security of the system and is fully accountable for any adverse impact to the organization if a breach of security occurs. Thus, responsibility and accountability are core principles that characterize accreditation.

3.8 POST-IMPLEMENTATION REVIEW

Projects should be formally closed to provide accurate information on project results, improve future projects and allow an orderly release of project resources. The closure process should determine whether project objectives were met or excused and should identify lessons learned to avoid mistakes and encourage repetition of good practices. In contrast to project closure, a post-implementation review typically is carried out in several weeks or months after project completion, when the major benefits and shortcomings of the solution implemented will be realized. The review is part of a benefits realization process and includes an estimate of the project's overall success and impact on the business.

A post-implementation review is also used to determine whether appropriate controls were built into the system. It should consider both the technical details and the process that was followed in the course of the project, including the following:

- Adequacy of the system:
  - Does the system meet user requirements and business objectives?
  - Have controls been adequately defined and implemented?

- Projected cost versus benefits or ROI measurements
- Recommendations that address any system inadequacies and deficiencies
- Plan for implementing any recommendations
- Assessment of the development project process:
  - Were the chosen methodologies, standards and techniques followed?
  - Were appropriate project management techniques used?
  - Is the risk of operating the system within acceptable risk levels?

Not all lessons associated with a given project may be immediately evident upon completion. It is good practice for the project development team and a selection of end users to perform a second, joint review after the project has been completed and the system has been in production for a sufficient time period to assess its effectiveness and value to the organization.

Closing a project is a formal process that focuses on capturing lessons learned for future use. **Figure 3.24** summarizes five steps that should be included in the closing of a project.

It is important to note that, for a post-implementation review to be effective, the information to be reviewed should be identified during the project feasibility and design phase, and collected during each stage of the project. For instance, the project manager might establish certain checkpoints to measure effectiveness of software development processes and accuracy of software estimates during the project execution. Business measurements should also be established up front and collected before the project begins and after the project is implemented (for examples of CSF measurements, see **figure 3.25**).

It is also important to allow a sufficient number of business cycles to be executed in the new system to realize the new system's actual ROI.

A post-implementation project review should be performed jointly by the project development team and appropriate end users. Typically, the focus of this type of internal review is to assess and critique the project process, whereas a post-implementation review has the objective of assessing and measuring the value the project has on the business. Alternatively, an

independent group not associated with the project implementation (internal or external audit) can perform a post-implementation review.

Figure 3.24—Project Closeout Steps	
Step	**Action**
1	Assign responsibility for any outstanding issues to specific individuals and identify the related budget for addressing these issues (if applicable).
2	Assign custody of contracts, and either archive documentation or pass it on to those who need it.
3	Conduct a post-implementation review with the project team, development team, users and other stakeholders to identify lessons learned that can be applied to future projects. Include the following information: • Content-related criteria, such as: – Fulfillment of deliverable targets and any additional objectives − Attainment of project-related incentives – Adherence to the schedule and costs • Process-related criteria, such as: – Team dynamics and internal communication – Relationships between the project team and external stakeholders
4	Document any risk that was identified in the course of the project, including risk that may be associated with proper use of the deliverables, and update the risk register.
5	Complete a second post-implementation review after the project deliverables have been completed for long enough to realize the true business benefits and costs, and use the results of this review to measure the project's overall success and impact on the business.

Figure 3.25—Measurements of Critical Success Factors	
Productivity	Dollars spent per user Number of transactions per month Number of transactions per user
Quality	Number of discrepancies Number of disputes Number of occurrences of fraud/misuse detection
Economic value	Total processing time reduction

	Monetary value of administration costs
Customer service	Turnaround time for customer question handling Frequency of useful communication to users

3.8.1 IS AUDITOR'S ROLE IN POST-IMPLEMENTATION REVIEW

An IS auditor performing a post-implementation review should be independent of the system development process. Therefore, an IS auditor involved in consulting with the project team on the development of the system should not perform this review. Unlike internal project team reviews, post-implementation reviews performed by an IS auditor tend to concentrate on the control aspects of the system development and implementation processes.

It is important that all audit involvement in the development project be thoroughly documented in the audit work papers to support an IS auditor's findings and recommendations. This audit report and documentation should be reused during maintenance and changes to validate, verify and test the impact of any changes made to the system. The system should periodically undergo a review to ensure the system is continuing to meet business objectives in a cost-effective manner and control integrity still exists.

Note: The CISA candidate should be familiar with issues related to dual controls as they apply to authorization within the post-implementation review and with those related to reviewing results of live processing.

An IS auditor should perform the following functions:

- Determine if the system's objectives and requirements were achieved. During the post-implementation review, careful attention should be paid to the end users' utilization, trouble tickets, work orders and overall satisfaction with the system. This will indicate whether the system's objectives and requirements were achieved.
- Determine if the cost benefits identified in the feasibility study are being measured, analyzed and accurately reported to management.

- Review program change requests performed to assess the type of changes required of the system. The type of changes requested may indicate problems in the design, programming or interpretation of user requirements.
- Review controls built into the system to ensure that they are operating according to design. If an EAM was included in the system, this module should be used to test key operations.
- Review operators' error logs to determine if there are any resource or operating problems inherent within the system. The logs may indicate inappropriate planning or testing of the system prior to implementation.
- Review input and output control balances and reports to verify that the system is processing data accurately.

CASE STUDY

Wonderwheels is a major national retailer specializing in outdoor sports, hunting, fishing and camping, including a wide variety of all-terrain vehicles (ATVs), which inspired its name. The company has operations currently based in the United States and has a long-term business plan to expand its retail centers to selected parts of the European Union (EU).

As part of ongoing current operations, management has asked an internal IS auditor to review the company's readiness for complying with requirements for protecting cardholder information. This is meant to be a high-level overview of where the firm stands and not a point-by-point review of its compliance with the specific standard (which would be undertaken as a separate engagement later in the year).

During the initial assessment, the IS auditor learned the following information:

- **Point-of-sale (POS) register encryption**—The retailer uses wireless POS registers that connect to application servers located at each store. These registers use wired equivalent protection (WEP) encryption.
- **POS local application server locations**—The POS application server, usually located in the middle of each store's customer service area, forwards all sales data over a frame relay network to database servers located at the retailer's corporate headquarters, with strong encryption applied to the data, which are then sent over a virtual private network (VPN) to the credit card processor for approval of the sale.
- **Corporate database locations**—Corporate databases are located on a protected screened subset of the corporate local area network.
- **Sales data distribution**—Weekly aggregated sales data, by product line, are copied as-is from the corporate databases to magnetic media and mailed to a third party for analysis of buying patterns.
- **Current ERP system compliance**—The current state of the company's ERP system is such that it may be out of compliance with newer laws and

regulations. During the initial assessment, the IS auditor determined that the ERP system does not adhere to the EU's General Data Protection Regulation (GDPR).

Additionally, Wonderwheels' database software has not been patched in over two years, due to a few factors:

- The vendor's support for the database package was dropped due to it being acquired by a competitor and refocusing the remaining business to other software services.
- Wonderwheels' management has implemented plans to upgrade to a new database package. The upgrade is underway; however, it is taking longer than anticipated.

Regarding the database upgrade, sizeable customizations were anticipated and are being carried out with a phased approach of partial deliverables. These deliverables are released to users for pilot usage on real data and actual projects. Concurrently, design and programming of the next phase are ongoing. In spite of positive initial test results, the internal audit group has voiced that it has not been included in key compliance decisions regarding the configuration and testing of the new system. In addition, operational transactions are often queued, or "hang" during execution, and more and more frequently, data are corrupted in the database. Additional problems have shown up—errors already corrected have started occurring again and functional modifications already tested tend to present other errors. The project, already late, is now in a critical situation.

1. Which of the following would present the **MOST** significant risk to the retailer?
   A. Database patches are severely out of date.
   B. Wireless POS registers use WEP encryption.
   C. Credit cardholder information is sent over the Internet.
   D. Aggregate sales data are mailed to a third party.

2. Based on the case study, which of the following controls would be the **MOST** important to implement?

A. POS registers should use two-factor authentication, with enforced complex passwords.
B. Wireless access points should use MAC address filtering.
C. The current ERP system should be patched for compliance with GDPR.
D. Aggregate sales data should be anonymized and encrypted prior to distribution.

3. In the preliminary report to management, regarding the state of the database upgrade, which of the following is **MOST** important for the IS auditor to include?
A. Internal audit should be included among the steering committee approvals.
B. There is a possibility that the new database may not be compatible with the existing ERP solution.
C. An ERP upgrade and/or patch is required in order to ensure updated database compatibility.
D. Internal audit should be able to review the upgraded database to ensure compliance with Payment Card Industry Data Security Standard (PCI-DSS).

4. In order to contribute more directly to help address the problems around the database upgrade, the IS auditor should:
A. Review the validity of the functional project specifications as the basis for an improved software baselining definition.
B. Propose to be included in the project team as a consultant for QC of deliverables.
C. Research the problems further to identify root causes and define appropriate countermeasures.
D. Contact the project leader and discuss the project plans and recommend redefining the delivery schedule using the PERT methodology.

ANSWERS TO CASE STUDY QUESTIONS

1. A. Unpatched database servers are located on a screened subnet; this would mitigate the risk to the organization.

   B. Use of WEP encryption would present the most significant risk because WEP uses a fixed secret key that is easy to break. Transmission of credit cardholder information by wireless registers would be susceptible to interception and would present a very serious risk.

   C. Sending credit cardholder data over the Internet would be less of a risk because strong encryption is being used.

   D. Because the sales data being sent to the third party are aggregate data, no cardholder information should be included.

2. A. According to the case study, it is unclear whether or not the POS registers already use two-factor authentication. It is known that aggregate sales data are copied onto other media as-is, without any controls, for external distribution.

   B. According to the case study, it is unclear whether or not the wireless access points use MAC address filtering. It is known that aggregate sales data are copied onto other media as-is, without any controls, for external distribution.

   C. Compliance with the GDPR, while important, is not the most important due to the current operations being only in the US, and the potential for expansion into the EU is a long-term vision for the company.

   D. It is unclear whether or not sales data are secure and free of personally identifiable information, such as credit card information and Social Security numbers. This would present the most significant risk and should be addressed.

3. **A. If internal audit is part of the steering committee, then it will have a say in the compliance and security-related controls to be included in production releases.**

   B. Ensuring database compliance is an operational responsibility and not an audit responsibility.

   C. Compatibility with existing architecture must be a function of the database implementation project team as a whole, which can

include internal audit and also includes operations. Therefore, it is not the best choice.

D. While it is important that the upgraded database solution be compliant with all regulations affecting the company, such a review should not be limited to one regulation. Therefore, it is not the best choice of those provided.

4. A. Functional project specifications should be executed by users and systems analysts, and not by the auditor.

B. To propose to be project consultant for quality would not bring about an essential contribution since quality is a formal characteristic, whereas in the current case, the problem is a substantial system instability.

C. The only appropriate action is additional research, even if the apparently technical nature of the problem renders it unlikely that the auditor may find it alone.

D. To contact the project leader and redesign the schedule of deliveries would not solve the problem. Furthermore, the definition of real causes may sensibly alter the project environm

Chapter 4:

Information Systems Operations and Business Resilience

Overview

Part A: Information Systems Operations

Part B: Business Resilience

4.12 Business Impact Analysis
4.13 System Resiliency
4.14 Data Backup, Storage and Restoration
4.15 Business Continuity Plan
4.16 Disaster Recovery Plans

Case Study

Case Study
Answers to Case Study Questions

OVERVIEW

Information systems operations and business resilience are important to provide assurance to users and management that the expected level of service will be delivered. Service level expectations are derived from the organization's business objectives. Information technology (IT) service delivery includes information systems (IS) operations, IT services and management of IS and the groups responsible for supporting them. Disruptions are also an often-unavoidable factor of doing business. Preparation is key to being able to continue business operations while protecting people, assets and reputation. Employing business resiliency tactics helps organizations address these issues and limit the impact.

This domain represents 23 percent of the CISA examination (approximately 34 questions).

DOMAIN 4 EXAM CONTENT OUTLINE

Part A: Information Systems Operations

1. Common Technology Components
2. IT Asset Management
3. Job Scheduling and Production Process Automation
4. System Interfaces
5. End-User Computing
6. Data Governance
7. Systems Performance Management
8. Problem and Incident Management
9. Change, Configuration, Release, and Patch Management
10. IT Service Level Management
11. Database Management

Part B. Business Resilience

1. Business Impact Analysis (BIA)

2. System Resiliency
3. Data Backup, Storage, and Restoration
4. Business Continuity Plan (BCP)
5. Disaster Recovery Plans (DRPs)

LEARNING OBJECTIVES/TASK STATEMENTS

Within this domain, the IS auditor should be able to:

- Evaluate the organization's ability to continue business operations. (T13)
- Evaluate whether IT service management practices align with business requirements. (T20)
- Conduct periodic review of information systems and enterprise architecture. (T21)
- Evaluate IT operations to determine whether they are controlled effectively and continue to support the organization's objectives. (T22)
- Evaluate IT maintenance practices to determine whether they are controlled effectively and continue to support the organization's objectives. (T23)
- Evaluate database management practices. (T24)
- Evaluate data governance policies and practices. (T25)
- Evaluate problem and incident management policies and practices. (T26)
- Evaluate change, configuration, release, and patch management policies and practices. (T27)
- Evaluate end-user computing to determine whether the processes are effectively controlled. (T28)
- Evaluate policies and practices related to asset life cycle management. (T33)

SUGGESTED RESOURCES FOR FURTHER STUDY

Hobbs, Martyn; IT Asset Management: *A Pocket Survival Guide*, IT Governance Publishing, USA, 2011.

ISACA, COBIT® 2019, USA, 2018, *www.isaca.org/cobit*

International Organization for Standardization/International Electrotechnical Commission, ISO/IEC 20000-1:2018, *Information technology—Service*

management—Part 1: Service management system requirements, Switzerland, 2018, *www.iso.org/standard/70636.html*

Mullins, Craig S.; *Database Administration: The Complete Guide to DBA Practices and Procedures, 2nd Edition*, Addison-Wesley Professional, USA, 2012

Snedaker, Susan; *Business Continuity & Disaster Recovery for IT Professionals 2nd Edition*, Syngress Publishing Inc., USA, 2013

Wallace, Michael; Lawrence Webber; *The Disaster Recovery Handbook; A Step-by-Step Plan to Ensure Business Continuity and Protect Vital Operations, Facilities, and Assets, 2nd Edition*, AMACOM, USA, 2010

SELF-ASSESSMENT QUESTIONS

CISA self-assessment questions support the content in this manual and provide an understanding of the type and structure of questions that typically appear on the exam. Often a question will require the candidate to choose the **MOST** likely or **BEST** answer among the options provided. Please note that these questions are not actual or retired exam items. Please see the section "About This Manual" for more guidance regarding practice questions.

4-1 Which one of the following provides the **BEST** method for determining the level of performance provided by similar information processing facility environments?

A. User satisfaction
B. Goal accomplishment
C. Benchmarking
D. Capacity and growth planning

4-2 For mission critical systems with a low tolerance to interruption and a high cost of recovery, the IS auditor, in principle, recommends the use of which of the following recovery options?

A. Mobile site
B. Warm site
C. Cold site
D. Hot site

4-3 Which of the following is the **MOST** effective method for an IS auditor to use in testing the program change management process?

A. Trace from system-generated information to the change management documentation
B. Examine change management documentation for evidence of accuracy
C. Trace from the change management documentation to a system-generated audit trail
D. Examine change management documentation for evidence of completeness

4-4 Which of the following would allow an enterprise to extend its intranet across the Internet to its business partners?

A. Virtual private network
B. Client-server
C. Dial-up access
D. Network service provider

4-5 The classification based on criticality of a software application as part of an IS business continuity plan is determined by the:

A. nature of the business and the value of the application to the business.
B. replacement cost of the application.
C. vendor support available for the application.
D. associated threats and vulnerabilities of the application.

4-6 When conducting an audit of client-server database security, the IS auditor should be **MOST** concerned about the availability of:

A. system utilities.
B. application program generators.
C. systems security documentation.
D. access to stored procedures.

4-7 When reviewing a network used for Internet communications, an IS auditor will **FIRST** examine the:

A. validity of password change occurrences.
B. architecture of the client-server application.
C. network architecture and design.
D. firewall protection and proxy servers.

4-8 An IS auditor should be involved in:

A. observing tests of the disaster recovery plan.
B. developing the disaster recovery plan.
C. maintaining the disaster recovery plan.
D. reviewing the disaster recovery requirements of supplier contracts.

4-9 Data mirroring should be implemented as a recovery strategy when:

A. recovery point objective (RPO) is low.
B. recovery point objective (RPO) is high.
C. recovery time objective (RTO) is high.
D. disaster tolerance is high.

4-10 Which of the following components of a business continuity plan is **PRIMARILY** the responsibility of an organization's IS department?

A. Developing the business continuity plan

B. Selecting and approving the recovery strategies used in the business continuity plan
C. Declaring a disaster
D. Restoring the IT systems and data after a disaster

ANSWERS TO SELF-ASSESSMENT QUESTIONS

4-1 A. User satisfaction is the measure to ensure that an effective information processing operation meets user requirements.
B. Goal accomplishment evaluates effectiveness involved in comparing performance with predefined goals.
C. Benchmarking provides a means of determining the level of performance offered by similar information processing facility environments.
D. Capacity and growth planning are essential due to the importance of IT in organizations and the constant change in technology.

4-2 A. Mobile sites are specially designed trailers that can be quickly transported to a business location or to an alternate site to provide a ready-conditioned information processing facility (IPF).
B. Warm sites are partially configured, usually with network connections and selected peripheral equipment—such as disk drives, tape drives and controllers—but without the main computer.
C. Cold sites have only the basic environment to operate an IPF. Cold sites are ready to receive equipment, but do not offer any components at the site in advance of the need.
D. Hot sites are fully configured and ready to operate within several hours or, in some cases, even minutes.

4-3 **A. When testing change management, the IS auditor should always start with system-generated information, containing the date and time a module was last updated, and trace from there to the documentation authorizing the change.**
B. Focusing exclusively on the accuracy of the documentation examined does not ensure that all changes were, in fact,

documented.

C. To trace in the opposite direction would run the risk of not detecting undocumented changes.

D. Focusing exclusively on the completeness of the documentation examined does not ensure that all changes were, in fact, documented.

4-4 **A. Virtual private network (VPN) technology allows external partners to securely participate in the extranet using public networks as a transport or shared private network. Because of low cost, using public networks (Internet) as a transport is the principal method. VPNs rely on tunneling/encapsulation techniques, which allow the Internet Protocol (IP) to carry a variety of different protocols (e.g., SNA and IPX).**

B. Client-server does not address extending the network to business partners (i.e., client-servers refers to a group of computers within an organization connected by a communications network where the client is the requesting machine and the server is the supplying machine).

C. Although it may be technically possible for an enterprise to extend its intranet using dial-up access, it would not be practical or cost effective to do so.

D. A network service provider may provide services to a shared private network by providing Internet services, but it does not extend an organization's intranet.

4-5 **A. The criticality classification is determined by the role of the application syste**m in supporting the strategy of the organization.

B. The replacement cost of the application does not reflect the relative value of the application to the business.

C. Vendor support is not a relevant factor for determining the criticality classification.

D. The associated threats and vulnerabilities will get evaluated only if the application is critical to the business.

4-6 **A. System utilities may enable unauthorized changes to be made**

to data on the client-server database. In an audit of database security, the controls over such utilities would be the primary concern of the IS auditor.

B. Application program generators are an intrinsic part of client-server technology, and the IS auditor would evaluate the controls over the generators access rights to the database rather than their availability.

C. Security documentation should be restricted to authorized security staff, but this is not a primary concern.

D. Access to stored procedures is not a primary concern.

4-7 A. Reviewing validity of password changes would be performed as part of substantive testing.

B. Understanding the network architecture and design is the starting point for identifying the various layers of information and the access architecture across the various layers, such as client-server applications.

C. The first step in auditing a network is to understand the network architecture and design. Understanding the network architecture and design provides an overall picture of the network and its connectivity.

D. Understanding the network architecture and design is the starting point for identifying the various layers of information and the access architecture across the various layers, such as proxy servers and firewalls.

4-8 **A. The IS auditor should always be present when disaster recovery plans are tested to ensure that the tested recovery procedures meet the required targets for restoration, that recovery procedures are effective and efficient, and to report on the results, as appropriate.**

B. IS auditors may be involved in overseeing plan development, but they are unlikely to be involved in the actual development process.

C. Similarly, an audit of plan maintenance procedures may be conducted, but the IS auditor normally would not have any

responsibility for the actual maintenance.

D. An IS auditor may be asked to comment upon various elements of a supplier contract, but, again, this is not always the case.

4-9 **A. Recovery point objective (RPO) is the earliest point in time at which it is acceptable to recover the data. In other words, RPO indicates the age of the recovered data (i.e., how long ago the data were backed up or otherwise replicated). If RPO is very low, such as minutes, it means that the organization cannot afford to lose even a few minutes of data. In such cases, data mirroring (synchronous data replication) should be used as a recovery strategy.**

B. If RPO is high, such as hours, then other backup procedures—such as tape backup and recovery—could be used.

C. A high recovery time objective (RTO) means that the IT system may not be needed immediately after the disaster declaration/disruption (i.e., it can be recovered later).

D. RTO is the time from the disruption/declaration of disaster during which the business can tolerate nonavailability of IT facilities. If RTO is high, slower recovery strategies that bring up IT systems and facilities can be used.

4-10 A. Members of the organization's senior management are primarily responsible for overseeing the development of the business continuity plan for an organization and are accountable for the results.

B. Management is also accountable for selecting and approving the strategies used for disaster recovery.

C. IT may be involved in declaring a disaster but is not primarily responsible.

D. The correct choice is restoring the IT systems and data after a disaster. The IT department of an organization is primarily responsible for restoring the IT systems and data after a disaster within the designated timeframes.