Chapter 3:

Information Systems Acquisition, Development and Implementation

Section One: Overview

Definition
Objectives
Task and Knowledge Statements
Suggested Resources for Further Study
Self-assessment Questions
Answers to Self-assessment Questions

Section Two: Content

3.1 Quick Reference
3.2 Benefits Realization
3.3 Project Management Structure
3.4 Project Management Practices
3.5 Business Application Development
3.6 Virtualization and Cloud Computing Environments
3.7 Business Application Systems
3.8 Development Methods
3.9 Infrastructure Development/Acquisition Practices
3.10 Information Systems Maintenance Practices
3.11 System Development Tools and Productivity Aids
3.12 Process Improvement Practices
3.13 Application Controls
3.14 Auditing Application Controls
3.15 Auditing Systems Development, Acquisition and Maintenance
3.16 Case Studies
3.17 Answers to Case Study Questions

Section One: Overview

DEFINITION

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.

OBJECTIVES

The objective of this domain is to ensure that the CISA candidate understands and can provide assurance that the practices for the acquisition, development, testing and implementation of information systems meet the organization's strategies and objectives.

This domain represents 18 percent of the CISA examination (approximately 27 questions).

TASK AND KNOWLEDGE STATEMENTS

TASKS

There are seven tasks within the domain covering information systems acquisition, development and implementation:

T3.1 Evaluate the business case for the proposed investments in information systems acquisition, development, maintenance and subsequent retirement to determine whether it meets business objectives.
T3.2 Evaluate IT supplier selection and contract management processes to ensure that the organization's service levels and requisite controls are met.
T3.3 Evaluate the project management framework and controls to determine whether business requirements are achieved in a cost-effective manner while managing risks to the organization.
T3.4 Conduct reviews to determine whether a project is progressing in accordance with project plans, is adequately supported by documentation and has timely and accurate status reporting.
T3.5 Evaluate controls for information systems during the requirements, acquisition, development and testing phases for compliance with the organization's policies, standards, procedures and applicable external requirements.
T3.6 Evaluate the readiness of information systems for implementation and migration into production to determine whether project deliverables, controls and organization's requirements are met.
T3.7 Conduct post-implementation reviews of systems to determine whether project deliverables, controls and organization's requirements are met.

KNOWLEDGE STATEMENTS

The CISA candidate must have a good understanding of each of the topics or areas delineated by the knowledge statements. These statements are the basis for the examination.

There are 14 knowledge statements within the domain covering information systems acquisition, development and implementation:

K3.1 Knowledge of benefits realization practices, (e.g., feasibility studies, business cases, total cost of ownership [TCO], return on investment [ROI])
K3.2 Knowledge of IT acquisition and vendor management practices (e.g., evaluation and selection process, contract management, vendor risk and relationship management, escrow, software licensing) including third-party outsourcing relationships, IT suppliers and service providers.
K3.3 Knowledge of project governance mechanisms (e.g., steering committee, project oversight board, project management office)
K3.4 Knowledge of project management control frameworks, practices and tools
K3.5 Knowledge of risk management practices applied to projects
K3.6 Knowledge of requirements analysis and management practices (e.g., requirements verification, traceability, gap analysis, vulnerability management, security requirements)
K3.7 Knowledge of enterprise architecture related to data, applications, and technology (e.g., web-based applications, web services, n-tier applications, cloud services, virtualization)
K3.8 Knowledge of system development methodologies and tools including their strengths and weaknesses (e.g., agile development practices, prototyping, rapid application development [RAD], object-oriented design techniques, secure coding practices, system version control)
K3.9 Knowledge of control objectives and techniques that ensure the completeness, accuracy, validity and authorization of transactions and data
K3.10 Knowledge of testing methodologies and practices related to the information system development life cycle (SDLC)
K3.11 Knowledge of configuration and release management relating to the development of information systems
K3.12 Knowledge of system migration and infrastructure deployment practices and data conversion tools, techniques and procedures
K3.13 Knowledge of project success criteria and project risk
K3.14 Knowledge of post-implementation review objectives and practices (e.g., project closure, control implementation, benefits realization, performance measurement)

Relationship of Task to Knowledge Statements

The task statements are what the CISA candidate is expected to know how to perform. The knowledge statements delineate each of the areas in which the CISA candidate must have a good understanding in order to perform the tasks. The task and knowledge statements are mapped in **figure 3.1** insofar as it is possible to do so. Note that although there is often overlap, each task statement will generally map to several knowledge statements.

Note: Hardware and software are integrated components of the technical infrastructure that make it possible for the business applications to operate. Considering them as separate audit items is an intrinsic scope limitation that requires a cautious definition of the audit objectives because these must generally be framed around their efficiency and effectiveness in supporting the business. The need to separately evaluate a component, or a class of components, must be determined as a part of the overall system assessment. A specific audit would be necessary, for example, if a preceding higher-level review concluded that a component, or a class of components, weakens the overall system performance. Furthermore, IS auditors must be aware of other synergies and restrictions that affect a specific component's functionality in the overall system because these have a direct impact on the significance of the audit results.

Figure 3.1—Task and Knowledge Statements Mapping

Task Statement	Knowledge Statements
T3.1 Evaluate the business case for the proposed investments in information systems acquisition, development, maintenance and subsequent retirement to determine whether it meets business objectives.	K3.1 Knowledge of benefits realization practices, (e.g., feasibility studies, business cases, total cost of ownership [TCO], return on investment [ROI]) K3.3 Knowledge of project governance mechanisms (e.g., steering committee, project oversight board, project management office) K3.5 Knowledge of risk management practices applied to projects K3.7 Knowledge of enterprise architecture related to data, applications, and technology (e.g., web-based applications, web services, n-tier applications, cloud services, virtualization)

	K3.13 Knowledge of project success criteria and project risk
T3.2 Evaluate IT supplier selection and contract management processes to ensure that the organization's service levels and requisite controls are met.	K3.2 Knowledge of IT acquisition and vendor management practices (e.g., evaluation and selection process, contract management, vendor risk and relationship management, escrow, software licensing) including third-party outsourcing relationships, IT suppliers and service providers.
T3.3 Evaluate the project management framework and controls to determine whether business requirements are achieved in a cost-effective manner while managing risks to the organization.	K3.1 Knowledge of benefits realization practices, (e.g., feasibility studies, business cases, total cost of ownership [TCO], return on investment [ROI]) K3.3 Knowledge of project governance mechanisms (e.g., steering committee, project oversight board, project management office) K3.4 Knowledge of project management control frameworks, practices and tools K3.5 Knowledge of risk management practices applied to projects K3.6 Knowledge of requirements analysis and management practices (e.g., requirements verification, traceability, gap analysis, vulnerability management, security requirements) K3.8 Knowledge of system development methodologies and tools including their strengths and weaknesses (e.g., agile development practices, prototyping, rapid application development [RAD], object-oriented design techniques, secure coding practices, system version control) K3.13 Knowledge of project success criteria and project risk
T3.4 Conduct reviews to determine whether a project is progressing in accordance with project plans is adequately supported by documentation and has timely and accurate status reporting.	K3.1 Knowledge of benefits realization practices, (e.g., feasibility studies, business cases, total cost of ownership [TCO], return on investment [ROI]) K3.3 Knowledge of project governance mechanisms (e.g., steering committee, project oversight board, project management office) K3.4 Knowledge of project management control frameworks, practices and tools K3.5 Knowledge of risk management practices applied to projects K3.8 Knowledge of system development methodologies and tools including their strengths and weaknesses (e.g., agile development practices, prototyping, rapid application development [RAD], object-oriented design techniques, secure coding practices, system version control) K3.13 Knowledge of project success criteria and project risk
T3.5 Evaluate controls for information systems during the requirements, acquisition, development and testing phases for compliance with the organization's policies, standards, procedures and applicable external requirements.	K3.2 Knowledge of IT acquisition and vendor management practices (e.g., evaluation and selection process, contract management, vendor risk and relationship management, escrow, software licensing) including third-party outsourcing relationships, IT suppliers and service providers. K3.4 Knowledge of project management control frameworks, practices and tools K3.5 Knowledge of risk management practices applied to projects K3.6 Knowledge of requirements analysis and management practices (e.g., requirements verification, traceability, gap analysis, vulnerability management, security requirements) K3.7 Knowledge of enterprise architecture related to data, applications, and technology (e.g., web-based applications, web services, n-tier applications, cloud services, virtualization) K3.8 Knowledge of system development methodologies and tools including their strengths and weaknesses (e.g., agile development practices, prototyping, rapid application development [RAD], object-oriented design techniques, secure coding practices, system version control) K3.9 Knowledge of control objectives and techniques that ensure the completeness, accuracy, validity, and authorization of transactions and data K3.10 Knowledge of testing methodologies and practices related to the information system development life cycle (SDLC) K3.11 Knowledge of configuration and release management relating to the development of information systems K3.12 Knowledge of system migration and infrastructure deployment practices and data conversion tools, techniques and procedures
T3.6 Evaluate the readiness of information systems for implementation and migration into production to determine whether project deliverables, controls and organization's requirements are met.	K3.5 Knowledge of risk management practices applied to projects K3.6 Knowledge of requirements analysis and management practices (e.g., requirements verification, traceability, gap analysis, vulnerability management, security requirements) K3.8 Knowledge of system development methodologies and tools including their strengths and weaknesses (e.g., agile development practices, prototyping, rapid application development [RAD], object-oriented design techniques, secure coding practices, system version control) K3.9 Knowledge of control objectives and techniques that ensure the completeness, accuracy, validity, and authorization of transactions and data K3.10 Knowledge of testing methodologies and practices related to the information system development life cycle (SDLC) K3.11 Knowledge of configuration and release management relating to the development of information systems K3.12 Knowledge of system migration and infrastructure deployment practices and data conversion tools, techniques, and procedures K3.13 Knowledge of project success criteria and project risk
T3.7 Conduct post-implementation reviews of systems to determine whether project deliverables, controls and organization's requirements are met.	K3.1 Knowledge of benefits realization practices, (e.g., feasibility studies, business cases, total cost of ownership [TCO], return on investment [ROI]) K3.4 Knowledge of project management control frameworks, practices and tools K3.9 Knowledge of control objectives and techniques that ensure the completeness, accuracy, validity and authorization of transactions and data K3.10 Knowledge of testing methodologies and practices related to the information system development life cycle (SDLC) K3.13 Knowledge of project success criteria and project risk K3.14 Knowledge of post-implementation review objectives and practices (e.g., project closure, control implementation, benefits realization, performance measurement)

Knowledge Statement Reference Guide

Each knowledge statement is explained in terms of underlying concepts and relevance of the knowledge statement to the IS auditor. It is essential that the exam candidate understand the concepts. The knowledge statements are what the IS auditor must know in order to accomplish the tasks. Consequently, only the knowledge statements are detailed in this section.

The sections identified in K3.1 through K3.14 are described in greater detail in section two of this chapter.

K3.1 Knowledge of benefits realization practices (e.g., feasibility studies, business cases, total cost of ownership [TCO], return on investment [ROI])

Explanation	Key Concepts	Reference in Manual
The premise of benefits realization is driven from a well-founded concern at board and senior management levels that the large expenditures on IT-related initiatives are not realizing the business benefits they promise. 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. Benefits realization of both systems and software projects are a compromise of major factors such as cost, quality, development/delivery time, reliability and dependability. Prior to initiating these types of projects, senior management performs a comprehensive study and evaluates which factors are "qualifying" or "winning." These results are then compared to strengths, weaknesses and competencies of services available to complete and maintain systems or software to be delivered by the project. Organizations should employ structured project management principles to support changes to their information systems environment. The IS auditor should understand how the business defines business cases, processes used during feasibility studies and resultant determinations with regard to ROI) for development-related projects. If companies fail to consistently meet their ROI objectives, this may suggest weakness in their system development life cycle (SDLC) and related project management practices.	Understanding the business case development approach for program management and SDLC processes	3.2 Benefits Realization 3.2.2 Business Case Development and Approval 3.2.3 Benefits Realization Techniques 3.5.2 Description of Traditional SDLC Phases 3.15.2 Feasibility Study
	Understanding benefits realization techniques	3.2.3 Benefits Realization Techniques

K3.2 Knowledge of IT acquisition and vendor management practices (e.g., evaluation and selection process, contract management, vendor risk and relationship management, escrow, software licensing) including third-party outsourcing relationships, IT suppliers and service providers

Explanation	Key Concepts	Reference in Manual
Based on the system or software requirements defined and feasibility studies, the project management team must evaluate if the organization will use internal development resources, a combination of internal resources supplemented by vendor-provided capabilities and products or completely outsource the project system/software development. Outsourced or vended capabilities, services and products must be evaluated for the risk these outside entities bring to the organization and the impact on benefits-realization goals. The IS auditor must understand the variety of vendor provided services (commercial off-the-shelf hardware/software products, outsourced services to include cloud offerings, managed services, etc.) and the functional requirements these services are addressing. Furthermore, the IS auditor needs to understand the vendors' service level agreements that are in place to address system/software operational and technical support requirements. Additional considerations also include suppliers' financial viability, licensing scalability and provisions for software escrow. Use of vendors can speed a project and potentially reduce total costs. However, use of vendors adds risk, especially if the vendor is a single- or sole-source provider. Proper vendor management can reduce or prevent problems caused when a vendor is chosen that is unable to achieve the required solution or time frame. Proper vendor management can also ensure that contracts address business needs and do not expose the business to unnecessary risk. Contract management processes may include many activities (a complete contract life cycle) and involve various organizational units (user department, IT development, finance, legal). Standards, frameworks and certifications play an important role because they are cost-effective ways of assessing and transmitting trust and they save effort, time and paperwork. The overall process may include specifications development, contract solicitation, vendor evaluation and selection, contract drafting, negotiation and execution, change orders, continuous vendor rating and monitoring, and contract close out. Although they are not legal or "contract auditors," IS auditors must understand the importance of requirements specification that forms the request for proposal (RFP). The IS auditor must understand the need for required security and other controls to be specified, the essential elements of vendor selection to ensure that a reliable and professional vendor is chosen and the essential contents of the contract—most notably, the need, as appropriate, for an escrow agreement to be in place. The right to audit must also be addressed in the contract.	Understanding system and software project acquisition evaluation and selection process, contract management, vendor risk and relationship management, escrow, software licensing	3.4.2 Project Planning 3.9 Infrastructure Development/ Acquisition Practices 3.9.4 Hardware Acquisition 3.9.5 System Software Acquisition
	Understanding the key information that should be included in a vendor contract	3.9.4 Hardware Acquisition 3.9.5 System Software Acquisition

K3.3 Knowledge of project governance mechanisms (e.g., steering committee, project oversight board, project management office)

Explanation	Key Concepts	Reference in Manual
All projects require governance structures, policies and procedures, and specific control mechanisms assure strategic and tactical alignment with the organization's respective goals and objectives and management of associated risk. Without proper governance oversight, all aspects of a project may be endangered. The IS auditor needs to understand program management governance concepts and how to evaluate the program office and/or project steering committee integration within the organization. With this understanding, the IS auditor can perform more proactive evaluations of top level system/software development activities within the organization at both an enterprise and operational level that can have dramatic impacts, both positive and negative, on the organization.	Understanding project portfolio management and project oversight structure and processes	3.2.1 Portfolio/Program Management 3.2.2 Business Case Development and Approval 3.4.4 Project Controlling 3.5.4 Risk Associated With Software Development

K3.4 Knowledge of project management control frameworks, practices and tools

Explanation	Key Concepts	Reference in Manual
Effective and efficient project management requires that the project manager's skill set be commensurate with the project at hand. Without the application of project management practices, tools and control frameworks, it is virtually impossible to manage all the relevant parameters of a large project. Projects need to 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. While the CISA exam will not test knowledge of any particular methodology, the IS auditor must understand the need for an established development management framework within the organization, the constituent elements of a standard methodology, and the contents and deliverables of each phase in order to ascertain the degree of necessary audit involvement.	Understanding project management practices, tools and control frameworks	3.3 Project Management Structure 3.3.1 General Aspects 3.3.3 Project Organizational Forms 3.3.5 Project Objectives 3.3.6 Roles and Responsibilities of Groups and Individuals 3.4 Project Management Practices 3.4.1 Initiation of a Project 3.4.2 Project Planning 3.4.3 Project Execution 3.4.4 Project Controlling 3.4.5 Closing a Project
	Understanding the system development life cycle (SDLC) process as it relates to project management	3.5.2 Description of Traditional SDLC Phases 3.8 Development Methods 3.8.1 Use of Structured Analysis, Design and Development Techniques 3.9 Infrastructure Development/ Acquisition Practices 3.12 Process Improvement Practices 3.12.1 Business Process Reengineering and Process Change Projects 3.12.2 ISO/IEC 25010:2011 3.12.3 Capability Maturity Model Integration 3.12.4 ISO/IEC 330xx Series

K3.5 Knowledge of risk management practices applied to projects

Explanation	Key Concepts	Reference in Manual
Risk is defined as a possible negative event or condition that would disrupt 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. The IS auditor needs to understand how risk management processes are integrated throughout program management processes and system and software development activities.	Understanding of the key concepts and processes used to mitigate risk with regards to program and project management	3.4.4 Project Controlling 3.7 Business Application Systems 3.15.1 Project Management

K3.6 Knowledge of requirements analysis and management practices (e.g., requirements verification, traceability, gap analysis, vulnerability management, security requirements)

Explanation	Key Concepts	Reference in Manual
Requirements capture is one of the most critical and difficult activities of the development life cycle. It is critical because it is an initial phase and the basis for every other activity. It is difficult because it involves subjective decisions and extensive interaction between units, functions and people (mainly between users and developers) with very different functions, interests and skills. Certain techniques and tools (interviews, protocol analysis, etc.) facilitate the requirements capture. Requirements should be prudent, feasible, cost-effective and aligned with an enterprise's strategy, plans and policies. Requirements should be duly documented to facilitate the understanding of the developers and formally approved and frozen (baselined) to prevent scope creep. It is incumbent that the IS auditor understand the life cycle of program, project and unique system and software development requirements. This understanding covers the processes of discovery of key functional, operational and security requirements through the maintenance, verification, traceability, gap analysis and vulnerability management controls implementation over requirements baseline.	Understanding the requirements analysis during SDLC phases	3.5 Business Application Development 3.5.1 Traditional SDLC Approach 3.5.2 Description of Traditional SDLC Phases 3.5.3 Integrated Resource Management Systems
	Understanding the use of the V-model to reduce the risk of not meeting requirements	3.5 Business Application Development
	Understanding the sponsor's role in developing success criteria	3.3.6 Roles and Responsibilities of Groups and Individuals
	Understanding the reasons to use an object breakdown structure	3.3.5 Project Objectives

K3.7 Knowledge of enterprise architecture related to data, applications and technology (e.g., web-based applications, web services, n-tier applications, cloud services, virtualization)

Explanation	Key Concepts	Reference in Manual
An enterprise architecture is a systematic description of an organization's structure, including business processes, information systems, human resources and organizational units. There are various enterprise architecture models; knowledge of any specific model is not essential for the IS auditor and will not be tested on the CISA exam. Enterprise architectures are supported or served by IT architectures (e.g., n-tier, client-server, web-based and distributed components). The IS auditor must understand the role of these components and how control objectives are met across all components to determine whether risk is sufficiently mitigated by these controls.	Understanding the components of an enterprise architecture	3.6 Virtualization and Cloud Computing Environments 3.7 Business Application Systems 3.7.1 E-commerce 3.7.16 Industrial Control Systems 3.9 Infrastructure Development/ Acquisition Practices

K3.8 Knowledge of system development methodologies and tools including their strengths and weaknesses (e.g., agile development practices, prototyping, rapid application development [RAD], object-oriented design techniques, secure coding practices, system version control)

Explanation	Key Concepts	Reference in Manual
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 new ways of organizing system and software projects that vary, or in some cases radically depart from, the traditional waterfall model. In addition, there has been continued evolution in the thinking about how best to analyze, design and construct software systems and in the information technologies available to perform these activities. The IS auditor's understanding of these differing methodologies enables them to better evaluate the existence and effectiveness of critical system development controls and to look at the selection of the methodologies and whether the process will properly address the project requirements (time, budget, resources). The CISA exam will not test the detailed knowledge of any specific methodology or project framework nor of any particular tool, but an understanding of the relative merits of and risk associated with recognized, standard tools is expected.	Understanding the different types of system/software development methodologies and tools	3.8 Development Methods
	How adopting these system/software development methodologies and tools impacts programs and respective projects	

K3.9 Knowledge of control objectives and techniques that ensure the completeness, accuracy, validity and authorization of transactions and data

Explanation	Key Concepts	Reference in Manual
Poor controls over data input, processing, storage or output increase the risk of loss to an enterprise. The IS auditor must be aware of the need for controls to ensure the authorization, accuracy and completeness of data input to, processing by and output from computer applications. The IS auditor also must know what types of control techniques are available at each level and how each may be evidenced in the form of reports, logs and audit trails.	Understanding authorization and editing, processing and output controls	3.7 Business Application Systems 3.13.1 Input/Origination Controls 3.13.2 Processing Procedures and Controls 3.13.3 Output Controls

K3.10 Knowledge of testing methodologies and practices related to the information system development life cycle (SDLC)

Explanation	Key Concepts	Reference in Manual
Integral to the system and software development project success is the proper selection of testing methodologies, development of testing plans fully traceable to requirements and 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. The IS auditor should also be aware of other forms of testing, such as unit, integration, sociability and regression testing. Furthermore, the IS auditor needs to understand the role quality assurance (QA) monitoring and evaluation play within quality of an enterprise'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).	Understanding types of testing in the system development process	3.14.6 Test Application Systems
	Understanding the goals of a quality assurance program within the context of the SDLC	3.3.6 Roles and Responsibilities of Groups and Individuals 3.5.2 Description of Traditional SDLC Phases
	Understanding quality assurance testing	3.14.4 Data Integrity Testing

K3.11 Knowledge of configuration and release management relating to the development of information systems

Explanation	Key Concepts	Reference in Manual
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 the 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, radio frequency [RF]). Knowledge of the current 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. The IS auditor should be aware of the tools available for managing configuration, change and release management and of the controls in place to ensure segregation of duties (SoD) between development staff and the production environment.	Understanding the configuration, change and release management processes and ways to examine the accomplishment of these processes	3.10 Information Systems Maintenance Practices 3.10.1 Change Management Process Overview 3.10.2 Configuration Management

K3.12 Knowledge of system migration and infrastructure deployment practices and data conversion tools, techniques and procedures

Explanation	Key Concepts	Reference in Manual
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 that they can be repeated on the production system. The IS auditor should ensure that any tools and techniques selected for the process are adequate and appropriate, that data conversion achieves the necessary objectives without data loss or corruption, and that any loss of data is both minimal and formally accepted by user management.	Understanding the criteria for successful data conversion during system migration	3.5.2 Description of Traditional SDLC Phases
	Understanding computer-aided software engineering and how it can be leveraged for data conversion	3.11.2 Computer-aided Software Engineering
	Understanding process improvement practices	3.12 Process Improvement Practices 3.12.1 Business Process Reengineering and Process Change Projects 3.12.2 ISO/IEC 25010:2011 3.12.3 Capability Maturity Model Integration 3.12.4 ISO/IEC 330xx Series

K3.13 Knowledge of project success criteria and project risk

Explanation	Key Concepts	Reference in Manual
From the start, program and project managers must identify and document the criteria that clearly indicate successfully meeting stakeholder requirements ranging from form, fit, function, schedule, budget and interoperability. As part of the program and project planning and overall management activities, risk to meeting the identified success criteria are also identified, analyzed and evaluated to enable managers to take preventive and corrective actions as needed. The IS auditor needs to understand the specific success criteria for the system/software programs and respective projects and how management is evaluating progress toward meeting these criteria along with addressing deficiencies that put the program and respective projects at risk.	Understanding methods to develop, monitor, measure and attain critical program and project success criteria	3.2.3 Benefits Realization Techniques 3.4 Project Management Practices 3.4.2 Project Planning 3.4.5 Closing a Project 3.15.8 Postimplementation Review

K3.14 Knowledge of post-implementation review objectives and practices (e.g., project closure, control implementation, benefits realization, performance measurement)

Explanation	Key Concepts	Reference in Manual
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 postimplementation 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.	Understanding the methods and benefits of postimplementation reviews following the completion of a project	3.2.3 Benefits Realization Techniques 3.4.5 Closing a Project 3.5.2 Description of Traditional SDLC Phases 3.15.8 Postimplementation Review

SUGGESTED RESOURCES FOR FURTHER STUDY

Bonham, Stephen S.; *IT Project Portfolio Management*, Artech House Inc., USA, 2005

Maizlish, Bryan; Robert Handler; *IT Portfolio Management Step-by-Step: Unlocking the Business Value of Technology*, John Wiley & Sons, USA, 2005

Natan, Ron Ben; *Implementing Database Security and Auditing*, Elsevier Digital Press, USA, 2005

Project Management Institute (PMI), *www.pmi.org*

Project Management Institute (PMI), *A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 5th Edition*, USA, 2013

Wysocki, Robert K.; *Effective Project Management: Traditional, Agile, Extreme, 6th Edition*, Wiley Publishing Inc., USA, 2011

Note: Publications in bold are stocked in the ISACA Bookstore.

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 have typically appeared on the exam. Questions are written in a multiple-choice format and designed for one best answer. Each question has a stem (question) and four options (answer choices). The stem may be written in the form of a question or an incomplete statement. In some instances, a scenario or a description problem may also be included. These questions normally include a description of a situation and require the candidate to answer two or more questions based on the information provided. Many times question will require the candidate to choose the **MOST** likely or **BEST** answer among the options provided.

In each case, the candidate must read the question carefully, eliminate known incorrect answers and then make the best choice possible. Knowing the format in which questions are asked, and how to study and gain knowledge of what will be tested, will help the candidate correctly answer the questions.

3-1 To assist in testing a core 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 (BPR), an IS auditor found that a key 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 (ERP) software used by a bank?

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, the IS auditor should:

A. assess the feasibility 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 (SDLC) would be replaced with:

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

3-8 User specifications for a project using the traditional system development life cycle (SDLC) 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. 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 (BPR), many nonvalue-added controls will be eliminated. This is good, unless they increase the business and financial risk.
D. The 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 matches 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. Avalidity 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.**
B. A lack of documentation may not be as serious as not having 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, the IS auditor would concentrate on the completeness of the specifications.

3-7 **A. With purchase packages becoming more common, the design and development phases of the traditional life cycle have become replaceable with selection and configuration phases. A request for proposal (RFP) 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 postimplementation remain unaltered.
C. The other phases of the SDLC such as feasibility study, requirements definition, implementation and postimplementation 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. Functionality issues in terms of business usability are out of scope and are, in this case, part of the development phase.
D. Depending on the chosen approach (traditional waterfall, rapid application development, etc.), these discrepancies might show up during user training or acceptance testing—whatever occurs first.

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.

Section Two: Content

3.1 QUICK REFERENCE

Quick Reference Review

Chapter 3 addresses the need for systems and infrastructure life cycle management. For an IS auditor to provide assurance that an enterprise's objectives are being met by the management practices of its systems and infrastructure, it is important to understand how an organization evaluates, develops, implements, maintains and disposes of its IT systems and related components. CISA candidates should have a sound understanding of the following items, not only within the context of the present chapter, but also to correctly address questions in related subject areas. It is important to keep in mind that it is not enough to know these concepts from a definitional perspective. The CISA candidate must also be able to identify which elements may represent the greatest risk and which controls are most effective at mitigating this risk. Examples of key topics in this chapter include the following:

- The management practice of benefits realization is the process by which an organization evaluates technology solutions to business problems. A feasibility study scopes the problem, outlines possible solutions and makes a recommendation. A business case is normally derived from the feasibility study and contains information for the decision-making process on whether it should be undertaken; if so, it becomes the basis for a project.
- A project portfolio is defined as all of the projects (related or unrelated) being carried out in an organization at a given point in time. A program can be seen as a group of projects that are linked together through common objectives, strategies, budgets and schedules. Portfolios, programs and projects are controlled by a project management office, which governs the processes of project management but are not typically involved in the management of their content.
- A project may be initiated from within any part of the organization, including the IT department. A project is time bound, with specific start and end dates, a specific objective and a set of deliverables. Throughout the project life cycle, the business case should be reviewed at predefined decision points. The project management processes of project initiation, planning, execution, controlling and project closing are management practices that outline how an organization works to implement projects.
- Using object and work breakdown structures, the project activities are identified and sized using a variety of techniques, such as estimating lines of code (LOC) to function and feature point analysis. The project manager then determines the optimal schedule by using the critical path method. The schedule can be depicted in several forms, such as the Gantt chart or Program Evaluation Review Technique (PERT) diagrams.
- A project must take into account the duration, costs and deliverables (quality) of the project. These elements are managed throughout the controlling and execution phases of the project. At project closing, a postproject review is conducted to assess lessons learned of the project management process.
- The management practices of specific IT processes are used for managing and controlling IT resources and activities. These IT processes form a system development life cycle (SDLC). A project may follow various forms of an SDLC, such as the classical waterfall method, the iterative method, the agile and verification and validation development models or other alternative methods of development, testing and implementation.
- The major phases of an SDLC include the release planning phase, definition phase, development phase, validation phase and deployment phase throughout which feasibility study, requirements definition, design, development, testing, training, implementation, certification and postimplementation review will be accomplished. The phases for each project depend on whether a solution is developed or acquired.
- Development processes may differ based on the use of system development tools and productivity aids including code generators, computer-aided software engineering (CASE) applications and collaborative code sharing and messaging services along with evolving fourth-generation languages and related utilities.
- It is essential to identify risk to the project. Business risk includes the new system not meeting the users' business requirements, whereas project risk relates to activities, such as the project exceeding the limits of the budget or time frame commonly caused by requirements and related scope creep.
- As part of its IT processes, an organization uses management practices to identify and analyze the needs for IT infrastructure. The physical architecture analysis, the definition of a new architecture and the necessary road map to move from one architecture to another is a critical task for an IT department.
- As an organization's systems and infrastructure change, management practices are used to manage that change (change management, release management and configuration management). Change management processes include the governance on how a change is submitted, assigned, tested, documented and approved.
- The IS auditor should understand the systems development, acquisition and maintenance methodology in use and to identify potential vulnerabilities and points requiring control. It is the IS auditor's role to advise the project team and senior management of the deficiencies and best practices within each of these processes.
- Other key impacts and influencers on an organization's management practices for its systems and architecture include e-commerce, business intelligence and cloud computing driven by market, industry and regulatory factors.
- In IT practices, input control procedures are evaluated to determine that they permit only valid, authorized and unique transactions. In an integrated application environment, controls are embedded and designed into the application that supports the processes. Business process control assurance involves evaluating controls at the process and activity level. These controls may be a combination of programmed and manual controls.

3.2 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.

The premise of benefits realization is that there is strong concern at board and senior management levels that the high expenditures on IT-related initiatives are not realizing the business benefits they promise. Studies and surveys also indicate high levels of loss from ill-planned and ill-executed initiatives. The focus on value has become more prevalent while the capability maturity of value management practices in most enterprises has remained low.

Benefits realization of projects 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, the IS auditor should understand how the business defines value or a return on investment (ROI) for development-related projects. If companies fail to consistently meet their ROI objectives, this may suggest weakness in their system development life cycle (SDLC) and related project management practices.

Example 1

Note: This is not an exhaustive example, but an illustrative example to explain the concept. Other factors may be applicable to each situation.

An entity 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. Costs
   1. Cost of developing the application

2. Cost of controls to ensure integrity of data at rest and in process, while ensuring nonrepudiation.

B. 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 the entity'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 comparing with the planned ROI.

3.2.1 PORTFOLIO/PROGRAM MANAGEMENT

A program can be seen as a group of projects and time-bound tasks that are closely linked together through common objectives, a common budget, and intertwined schedules and strategies. 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 system, organizational structure, and business processes.

The objective of program management is the successful execution of programs including, but not limited to, management of:
- Program scope, program financials (costs, resources, cash flow, etc.), program schedules, and program objectives and deliverables
- Program context and environment
- Program communication and culture
- Program 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, the program manager and the 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 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.

A project portfolio is defined as all the projects being carried out in an organization at a given point in time (snapshot). 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 the company'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 project management office and project portfolio groups. Specific integrative tools such as project management guidelines, standard project plans and project management marketing instruments are employed.

The project management office, 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 project management office is to improve project and program management quality and secure project success, but it can only focus on activities and tasks and not on project or program content. Therefore, an IS auditor has 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

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

Example 2

An entity is carrying out the following activities:

A. An entity 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.
   1. The entity is using its 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. The entity is using vendors to code the non-strategic business processes of procure to pay and financial accounting.

B. The entity 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 non-core 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.

3.2.2 BUSINESS CASE DEVELOPMENT AND APPROVAL

An important consideration in any IT project, whether it is the development of a new system or an investment in new infrastructure, is the business case. It has been increasingly recognized that the achievement of business benefits should drive projects.

A business case provides the information required for an organization to decide whether a project should proceed. Depending on the organization and often on the size of the investment, the development of a business case is either the first step in a project or a precursor to the commencement of a project.

The 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, meets requirements within established budgets and schedule requirements. The feasibility study will normally include the following six elements:

1. The **project scope** defines the business problem and/or opportunity to be addressed. It should be clear, concise and to the point.
2. The **current analysis** defines and establishes 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** are defined 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. The **approach** is the recommended system and/or software solution to satisfy the requirements. 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** is 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. A formal **review** of the feasibility study report is conducted with all stakeholders. This review will 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. 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 lessons learned list for use in future project studies.

Part of the work in developing options is to calculate and outline a business case for each solution in order to allow a comparison of costs and business benefits.

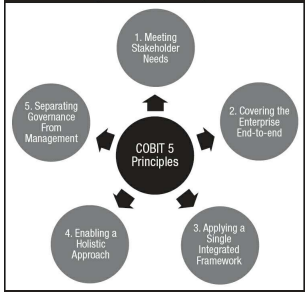
The business case should be of sufficient detail to describe the justification for setting up and continuing a project. The business case should provide the reasons for the project and answer the question, “Why should this project be undertaken?”

The business case should also 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 be no longer be valid, through increased costs or reduction in the anticipated benefits, the project sponsor or IT steering committee should consider whether the project should proceed. In a well-planned project, there will be decision points, sometimes called “stage gates” or “kill points,” at which the 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.

3.2.3 BENEFITS REALIZATION TECHNIQUES

COBIT 5 provides the industry accepted framework under which IT governance goals and objectives are derived from stakeholder drivers with the intent of enterprise IT generating business value from IT-enabled investments (i.e., achieve strategic goals and realize business benefits through effective and innovative use of IT). As indicated in **figure 3.2**, principle 1, Meeting Stakeholder Needs, stresses the need for enterprise IT to create value for stakeholders by maintaining a balance between the realization of benefits and the optimization of risk and use of resources. COBIT 5 addresses the required processes and other enablers to support business value creation through the use of IT. Because every enterprise has different objectives, an enterprise can customize COBIT 5 to suit its own context through the goals cascade—translating high-level enterprise goals into manageable, specific, IT-related goals and mapping these to specific processes and practices.

Figure 3.2—COBIT 5 Principles



Source: ISACA, COBIT 5, USA, 2012

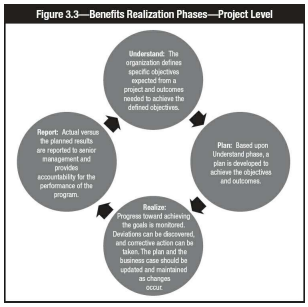
The majority of business benefits are obtained through changes enabled by technology. With the evolution of the application of IT from straight automation of work, through information management to business transformation, the realization of benefits has become a more complex task. Benefits do not just happen when the new technology is delivered; they occur throughout the business cycle.

A planned approach to benefits realization is required, looking beyond project cycles to longer term cycles that consider the total benefits and total costs throughout the life of the new system. Benefits rarely come about exactly as envisioned in plans. Organizations have to keep checking and adjusting strategies. This concept is called benefits management or benefits realization, and key elements are as follows:

- 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 encompass four phases, as indicated in the **figure 3.3**.

Figure 3.3—Benefits Realization Phases—Project Level



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

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. Enterprisewide benefits realization studies should be aggregated and lessons learned should fine-tune the enterprise benefit realization process.

Benefits realization often includes a postimplementation review six to eighteen months 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.

Benefits realization must be part of governance and management of projects. There must be business sponsorship of projects. 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 benefits.

3.3 PROJECT MANAGEMENT STRUCTURE

Today, many approaches to project management exist. Some are focused on software development, others have a more general approach; some concentrate on a holistic and systemic view, others provide a very detailed workflow including templates for document creation. Some of the most prominent *de facto* standards and organizations are the Project Management Body of Knowledge (PMBOK®) (i.e., IEEE standard 1490 from the Project Management Institute [PMI], Projects in a Controlled Environment [PRINCE2™] from the Office of Government Commerce (OGC) in the UK, and the International Project Management Association [IPMA]). Because there are significant differences in scope, content and wording in each of these standards, an IS auditor has to become familiar with the standard in use prior to involvement in specific projects.

Although each project management approach has its own pros and cons, several elements are common across all project management methodologies.

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

3.3.1 GENERAL ASPECTS

IS projects may be initiated from within any part of the organization, including the IT department.

A project is always a time-bound effort. A project can be complex and involve an element of risk. A project has specific objectives, deliverables and start and end dates. Most IS projects are divisible into explicit phases (e.g., SDLC). A project can be perceived as a group of complex tasks as well as a social system and/or a temporary organization (in contrast to the relatively stable structures of the permanent organization).

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.

The complexity of project management requires a careful and explicit design of the project management process, which is normally done for a business process but sometimes neglected for the project management process. Thus, all design issues applicable for business process engineering should also be applied for the project management process.

3.3.2 PROJECT CONTEXT AND ENVIRONMENT

A project context can be divided into a time and a social context. In the analysis of the content's dimension of the context, the following must be taken into account:

- Importance of the project in the organization
- Connection between the organization's strategy and the project
- Relationship between the project and other projects within the same program as well as other projects under different program(s)
- Connection between the project and the underlying business case

Because there are normally several projects running at the same time, the relationships between those projects have to be investigated to identify common objectives for the business organization, identify and manage risk, and identify resource connections. A common approach to address these issues is to establish a project portfolio management and/or a program management structure.

A project, by definition, has a specific start date and end date. The project's time context, however, should consider the phase before project startup and the phase after project closing. During the preproject phase, key activities and objectives that must be considered in project planning are identified. A thorough transfer of information about the negotiations and decisions as well as necessary knowledge from the preproject phase is critical. The expectations in regard to the postproject phase also influence the tasks to be fulfilled and the strategy for designing the project environment relationships.

Because a project represents a social system, it is also necessary to consider its relationships to its own social environments. The design of the project environment relationships is a project management activity. The objective is to determine all relevant environments for the project, which will have a significant influence on overall project planning and project success.

3.3.3 PROJECT ORGANIZATIONAL FORMS

Three major forms of organizational alignment for project management within the business organization can be observed:

1. Influence project organization
2. Pure project organization
3. Matrix project organization

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

In a pure project organization, the project manager has formal authority over those taking part in the project. Often, this is bolstered by providing a special working area for the project team that is separated from their normal office space.

In a matrix project organization, management authority is shared between the project manager and the department heads.

For an IS auditor, it is important to understand these organizational forms and their implications on controls in project management activities.

Requests for major projects should be submitted to and prioritized by the IT steering committee. A project manager should be identified and appointed by the IT steering committee. 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. IS auditors may be included in the project team as control experts. They may also provide an independent, objective review to ensure that the level of involvement (commitment) of the responsible parties is appropriate. In such cases, IS auditors are not performing an audit, but are participating on the project in an advisory role; depending on the level of their involvement, they 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 3.4**.

3.3.4 PROJECT COMMUNICATION AND CULTURE

Depending on the size and complexity of the project and the affected parties, communication when initiating the project management process may be achieved by:

- One-on-one meetings
- Kick-off meetings
- Project start workshops
- A combination of the three

One-on-one meetings and a project start workshop help to facilitate two-way communication between the project team members and the project manager. A kick-off meeting may be used by the 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.).

A preferred method to ensure that communication is open and clear among the project team 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.

As an independent social system, each project has its own culture that defines its norms and rules of engagement. The project culture cannot be described but manifests itself in the applied project management techniques, including project planning, forms of communication, etc. The dynamics of the project culture provide for a common understanding of what is seen as desirable and serves as an orientation for the team.

A project culture is comprised of shared norms, beliefs, values and assumptions of the project team. A key success factor for establishing the correct project culture is defining and adapting the unique characteristics of a project, which enables the correct culture to flourish that is a match to complexity of that project.

Methods for developing a project culture include the establishment of a project mission statement, project name and logo, project office or meeting place, project intranet, project team meeting rules and communication protocol, and project-specific social events.

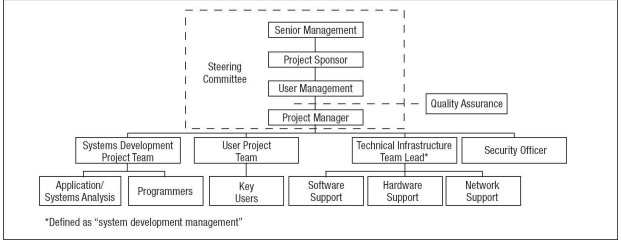
3.3.5 PROJECT OBJECTIVES

Project objectives are the specific action statements that support the road map to obtain established 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 objectives. When a project objective is defined, it always starts with an action verb. This syntax ensures that the objective is measurable and can be used to verify that the project outcome is directly traceable to the action of the objective.

A project needs clearly defined results that are specific, measurable, attainable, realistic and timely (SMART). A comprehensive project view ensures the consideration and consolidation of all closely coupled objectives. These objectives are broken down into main objectives, additional objectives and nonobjectives.

- **Main objectives** are the primary reason for the project and will always be directly coupled with business success (see section 3.5 Business Application Development).
- **Additional objectives** are objectives that are not directly related to the main results of the project but may contribute to project success (e.g., business unit reorganization in a software development project).

Figure 3.4—Project Management, Sample Organization Chart



- **Nonobjectives** are the results that are not to be expected on completion of the project. Defining the nonobjectives brings clarity to scope, and project boundaries become clearer. Nonobjectives shape the confines of the project deliverables and help all parties to gain a clear understanding of what results are within expectations and clearly not within scope of the project to avoid any ambiguities.

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 nontangible project results such as organizational development, to ensure that a material deliverable is not overlooked.

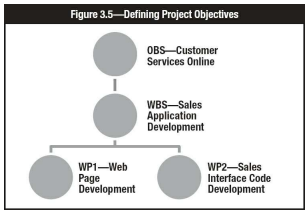
After the OBS has been compiled or a solution is defined, a work breakdown structure (WBS) is designed to structure all the tasks that are necessary to build up the elements of the OBS during the project. The WBS represents the project in terms of manageable and controllable units of work, serves as 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 between 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 nonobjectives. 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.**

Figure 3.5—Defining Project Objectives

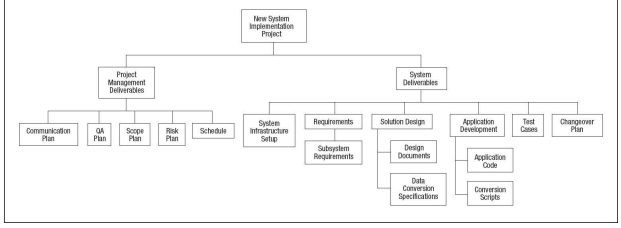


Key things to remember with WBS and respective WPs include:
- The top WBS level represents the final deliverable or project.
- Sub-deliverables contain work packages that are assigned to an organization's department or unit.
- All elements of the WBS do not need to be defined to the same level.
- WPs define the work, duration, and costs for the tasks required to produce the sub-deliverable.
- WPs should not exceed duration of 10 days.
- WPs need to be independent of each other in the WBS.
- WPs are unique; do not duplicate a WP 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 work packages 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 3.2.2 Business Case Development and Approval.

Figure 3.6—Sample Work Breakdown Structure



3.3.6 ROLES AND RESPONSIBILITIES OF GROUPS AND INDIVIDUALS

To achieve a successful completion and implementation of any new system, it is advisable that the audit function has an active part, where appropriate, in the life cycle development of the 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 e-commerce systems).

Additionally, there are other key stakeholders who should be involved in the system's design, development and implementation. The various roles and responsibilities of groups/individuals that may be involved in the development process are summarized as follows:
- **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.
- **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 management is concerned particularly with the following questions:
  – Are the required functions available in the software?
  – How reliable is the software?
  – How efficient is the software?
  – Is the software easy to use?
  – How easy is it to transfer or adapt old data from preexisting software to this environment?
  – How easy is it to transfer the software to another environment?
  – Is it possible to add new functions?
  – Does it meet regulatory requirements?
- **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. Each member must have the authority to make decisions related to system designs that will affect their respective departments. Generally, a project sponsor who would assume the overall ownership and accountability of the project will chair the steering committee. The project manager should also be a member of this committee. The project steering committee performs the following functions:
  – Reviews project progress regularly (for example, 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. The committee should evaluate progress and take action or make recommendations regarding personnel changes on the project team, managing budgets or schedules, changes in project objectives, and the need for redesign. The committee should be available to address risk and issues that are escalated and cannot be resolved at the project level. The project manager should have the ability to escalate such matters and rely on the project steering committee to resolve risk and project issues for the benefit of the organization. In some cases, the committee may recommend that the project be halted or discontinued.
- **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. It is crucial that success is translated to measurable and quantifiable terms. 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 that the application will support.
- **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.
- **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 also facilitate the definition of the scope of the project, manage the budget of the project, and control the activities via a project schedule. This person can be an end user, a member of the systems development team or a professional project manager. Where projects are staffed by personnel dedicated to the project, the project manager will have a line responsibility for such personnel.

- **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.
- **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.
- **Security officer**—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 (International Organization for Standardization [ISO]/International Electrotechnical Commission [IEC] 21827:2008 specifies the Systems Security Engineering–Capability Maturity Model® [SSE-CMM®]). 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.
- **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 (see section 3.12.4 ISO/IEC 330xx Series). Specific objectives of the QA function include:
  - Ensuring the active and coordinated participation by all relevant parties in the revision, evaluation, dissemination, and application of standards, management guidelines and procedures
  - Ensuring compliance with the agreed-on systems development methodology
  - Reviewing and evaluating large system projects at significant development milestones and making appropriate recommendations for improvement
  - Establishing, enhancing and maintaining a stable, controlled environment for the implementation of changes within the production software environment
  - Defining, establishing and maintaining a standard, consistent and well-defined testing methodology for computer systems
  - Reporting to management on systems that are not performing as defined or designed

It is essential for the IS auditor to understand the systems development, acquisition and maintenance methodology in use and to identify potential vulnerabilities and points requiring control. If controls are lacking (either as a result of the organizational structure or of the software methods used) or the process is disorderly, it is the 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 development and acquisition activities of appropriate controls or processes to implement and follow.

Note: The CISA candidate should be familiar with general roles and responsibilities of groups or individuals involved in the systems development process.

3.4 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 knowledge and practices are best described in terms of their component processes of initiating, planning, executing, controlling and closing a project. Overall characteristics of successful project planning are that it is a risk-based management process and iterative in nature. Project management techniques also provide systematic quantitative and qualitative approaches to software size estimating, scheduling, allocating resources and measuring productivity.

There are numerous project management techniques and tools 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. These tools typically provide assistance in areas described in the following sections.

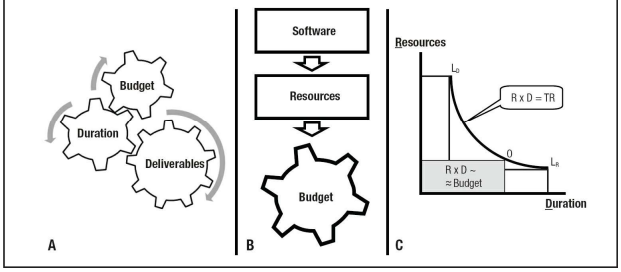
There are various elements of a project that should always be taken into account. Their relationship is shown in **figure 3.7**.

Project management should pay attention to three key intertwining elements: deliverables, duration and budget (**figure 3.7A**). 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 3.7B**) 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 3.7C** 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. If we choose 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.

Figure 3.7—Relationships Between Project Management Elements



3.4.1 INITIATION OF A PROJECT

A project will be initiated by a project manager or sponsor gathering the information required to gain approval for the project to be created. This will often be 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.

3.4.2 PROJECT PLANNING

System and software development/acquisition or maintenance projects have to be planned and controlled.

The project manager needs to determine:
- The various tasks that need to be performed to produce the expected business application system
- The sequence or the order in which these tasks need to be performed
- The duration or the time window for each task
- The priority of each task
- The IT and non-IT supporting resources, which 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

Realistically, on other than small projects, detailed task-level planning can only occur toward the end of a phase or iteration for the upcoming phase or iteration, or through to the next decision gate. Until work products and decisions are produced during the current phase or iteration, insufficient information is available to understand in detail what will occur in the next phase/iteration.

There are some techniques that are useful in creating a project plan and monitoring its progress throughout the execution of the project to provide continual support for making it successful. Some of these techniques are addressed in the following sections.

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

System Development Project Cost Estimation
Normally much larger in scope and size, the 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 a system 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 to the allocation of resources and to judge the time and cost required for its development, and to compare the total effort required by the resources. See **figure 3.7**.

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 has evolved over the years to become a multiple-point technique widely 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 3.8**) 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 takes into account 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. It is based on the number and complexity of inputs, outputs, files, interfaces and queries.

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

FPA Feature Points
In most standard applications, lists of functions are identified and the corresponding effort is estimated. In web-enabled applications, the development effort depends on the number of screens (forms), number of images, type of images (static or animated), features to be enabled, interfaces and cross-referencing that is required. Thus, from the point of view of web applications, the effort would include all that is mentioned under function point estimation, plus the features that need to be enabled for different types of user groups. The measurement would involve identification or listing of features, access rules, links, storage, etc.

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 toward 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)

Figure 3.8—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.

- 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:
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 consequence of software size estimation. This is a necessary step in properly scoping a project.

Alternatively, there are automated techniques for cost estimation of projects at each phase of system development. To use these products, a 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 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:
- 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 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.

Critical Path Methodology

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 where 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 that the activity will consume. There are different ways to estimate the activity duration.

Regardless of how the activity duration is estimated, the critical path is the sequence of activities whose sum of activity time is longer than that for any other path through the network. 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, their duration gives the shortest possible completion time for the overall project. Activities that are not in the critical path have time slack. This 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, and conversely, activities with zero slack time are on a critical path.

The critical path(s) and the slack times for a project are computed by simply working forward through the network (forward pass), computing the earliest possible completion time for each activity, until the earliest possible completion time for the total project is found. Then by working backward through the network, the latest completion time for each activity is found, the slack time computed and the critical path identified. This procedure is computer-supported for easy calculation and what-if scenarios.

Within limits, activities can be "crashed" (reduced in time by payment of a premium for early completion) or relaxed (with associated cost reductions). In this way, total duration and budget can be managed (within limits).

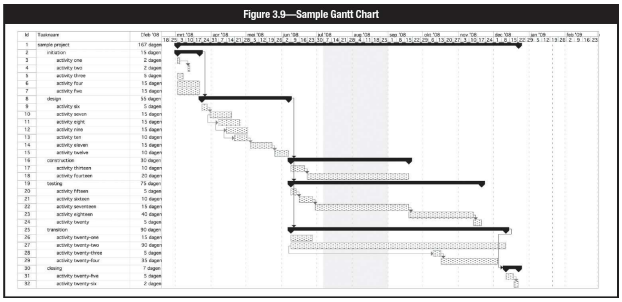
Most CPM packages facilitate the analysis of resource utilization per time unit (e.g., day, week, etc.) and resource leveling, which is a way to level off resource peaks and valleys. Resource peaks and valleys are expensive due to management, hiring, firing, and/or overtime and idle resource costs. A 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.

Gantt Charts

Gantt charts (**figure 3.9**) can be constructed to 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 be in progress concurrently and which activities must be completed sequentially. Gantt charts can also reflect the resources assigned to each task and by what percent allocation. The charts aid in identifying activities that have been completed early or late by comparison to a baseline. Progress of the entire project can be read from the Gantt chart to determine whether the project is behind, ahead or on schedule compared to 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.

Figure 3.9—Sample Gantt Chart



Program Evaluation Review Technique

PERT is a CPM-type technique which uses three different estimates of each activity duration in lieu of using a single number for each activity duration (as used by CPM). The three estimates are then reduced (applying a mathematical formula) to a single number 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 use of the PERT network management technique is shown in **figure 3.10**, where 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 most optimistic time (if everything went well) and the third is the pessimistic or worst-case 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:

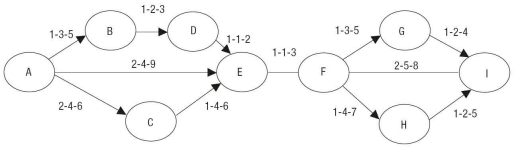
[Optimistic + Pessimistic + 4(most likely)]/6

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 is shortened (accelerated) or lengthened (delayed). In **figure 3.10**, the critical path is A, C, E, F, H and I.

The advantage of PERT over CPM is that the formula above 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.

Figure 3.10—PERT Network-based Chart



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 more 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 or time window. The project manager has some degree of flexibility and uses discretion in scoping requirements. 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 are normally performed together.

3.4.3 PROJECT EXECUTION

Once planning efforts have been completed, the program manager, in coordination with the program office, starts the actual execution of the planned tasks as described in the OBS, WBS and WPs (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.

3.4.4 PROJECT CONTROLLING

The controlling activities of a project include management of scope, resource usage and risk. It is important that new requirements for the project are 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.

Management of Scope Changes

Managing the scope of projects requires careful documentation in the form of a work breakdown structure. This documentation forms part of the project plan or the project baseline. For complex deliverables, it is best to document the WBS in a component management database (CMDB). 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. This process starts with a formal change request that contains a clear description of the requested change and the reasons for the change. Change requests must be submitted to the project manager. Obviously, 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. It is not sufficient to only monitor actual spending. Every budget and project plan presupposes a certain "productivity" of resources (e.g., if a task is planned to take 24 man-hours, then it is implicitly supposed that the resource being deployed is capable of finishing that task in 24 man-hours and, at the same time, delivering results at a satisfactory quality level). Whether this is actually happening can be checked with a technique called earned value analysis (EVA).

EVA consists of comparing the following metrics at regular intervals during the project: budget to date, actual spending to date, estimate to complete and estimate at completion. 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. This is according to budget, but the following question must be answered: Is this project on track? The answer cannot be known unless the estimate to complete is known. If productivity of the resource is as it is supposed to be, the resource would need another 16 hours to complete the task. Rather than assuming this to be the case, the worker must be asked how much more time is needed to complete the task. The answer might well be that another 22 hours are needed to complete the task. The estimate at completion is then 30 hours for the project. This results in an overrun of 25 percent. On the first day of this small project, the resource spent is according to budget, but the "earned value" of the eight hours spent is only two hours (at planned productivity). Obviously, reporting via time sheet and management are at the basis of all of these processes.

Management of Risk

Risk is defined as a possible negative event or condition that would disrupt 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.

The risk management process consists of five steps that are repeatedly executed during a project. Phase-end milestones are a good anchor point in time at which to review and update the initial risk assessments and the related mitigations. The risk management process steps are:
- **Identify risk**—Perform a brainstorming session with the team and create an inventory of possible risk.
- **Assess and evaluate risk**—Quantify the likelihood (expressed as a percentage) and the impact of the risk (expressed as an amount of money). The "insurance policy" that needs to be in the project budget is calculated as the likelihood multiplied by the impact.
- **Manage risk**—Create a risk management plan, describing the strategy adopted and measures to deal with the risk. Generally, the more important the risk, the more budget should be made available for countermeasures. Countermeasures include prevention, detection and damage control/reconstruction activities. Any risk can be mitigated, avoided, transferred or accepted depending on its severity, likelihood and cost of countermeasures and the enterprise's policy.
- **Monitor risk**—Discover risk that materializes, and act accordingly.
- **Evaluate the risk management process**—Review and evaluate the effectiveness and costs of the risk management process.

3.4.5 CLOSING A PROJECT

A project should have a finite life so, at some point, it is closed and the 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 areas to consider include:
- When will the project manager issue the final project closure notification?
- Who will the final project notification come from?
- 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?

Custody of contracts may need to be assigned, and documentation archived or passed on to those who will need it. It is also common at this stage to survey the project team, development team, users and other stakeholders to identify any lessons learned that can be applied to future projects including content-related criteria such as performance fulfillment, fulfillment of additional objectives, adherence to the schedule and costs, and process-related criteria such as quality of the project teamwork and relationships to relevant environments.

Review may be done in a formal process such as a postproject review in which lessons learned and an assessment of project management processes used are documented to allow reference, in the future, by other project managers or users working on projects of similar size and scope.

A postimplementation 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.

Note: Project management practice descriptions and related concepts and theories behind best practices have been brought together in "body of knowledge" (BoK) reference libraries. Certification schemes have subsequently been based upon such BoKs. There are three relevant professional organizations that have such certification schemes: PMI, IPMA and PRINCE2 Foundation.

3.5 BUSINESS APPLICATION DEVELOPMENT

Companies often commit significant IT resources (e.g., people, applications, facilities and technology) to develop, acquire, integrate and maintain application systems that are critical to the effective functioning of key business processes. These systems, in turn, often control critical information assets and should be considered assets that need to be effectively managed and controlled. IT processes for managing and controlling these IT resources and other such activities are part of a life cycle process with defined phases applicable to business application development, deployment, maintenance and retirement. In this process, each step or phase in the life cycle is an incremental step that lays the foundation for the next phase, for effective management control in building and operating business application systems.

The implementation process for business applications follows the project planning and management processes as outlined previously. Normally, the business application development project begins when an individual application feasibility study is 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 of these situations are tightly coupled with key business drivers. Key business drivers, in this context, can be defined as the attributes of a business function that drive the behavior and implementation of that business function to achieve the strategic business goals of the company.

Thus, all critical business objectives (as a breakdown of the corporate strategy) have to be translated into key business drivers for all parties involved in business operations during an SDLC project. Objectives should follow the SMART quality guidelines mentioned in section 3.3.5 Project Objectives, so that general requirements will be expressed in scorecard form, which allows objective evidence to be collected in order to measure the business value of an application and to prioritize requirements.

Benefits of using this approach are 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., costversus functionality) and mutually dependent key business drivers can be detected and resolved in early stages of an SDLC 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 the organization. All internal and external factors affected by the problem and their effect on the corporation should be identified.

A risk in any software development project is that the final outcome may not meet all requirements. Problems due to translation errors arise when initially defining the requirements for interim products. The 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. The verification and validation model, sometimes called the V-model, also emphasizes the relationship between development phases and testing levels (**figure 3.11**). 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 user-acceptance testing references the requirements.

From an IS auditor's perspective, the V-model's defined life cycle phases and specific points for review and evaluation provides the following advantages:
- The 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 auditor involvement.
- The 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 company procedures.
- The 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.
- The IS auditor can provide an evaluation of the methods and techniques applied through the development phases of the business application life cycle.

Any business application system developed will fall under one of two major categories:
- **Organizationcentric** (management information system [MIS], ERP, customer relationship management [CRM], supply chain management [SCM], etc.)—The objective of organizationcentric 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 organizationcentric applications. Organizationcentric application projects usually use the SDLC or other more detailed software engineering approaches for development.
- **End-user-centric computing**—The objective of an end-user-centric application is to provide different views of data for their performance optimization. This objective includes DSS, geographic information systems (GIS), techniques, etc. Most of these applications are developed using alternative development approaches.

3.5.1 TRADITIONAL SDLC APPROACH

Over the years, business application development has occurred largely through the use of the traditional SDLC phases shown in **figure 3.12**. Also referred to as the waterfall technique, this life cycle approach is the oldest and most widely used for developing business applications. The approach is based on a systematic, sequential approach to software development (largely of business applications) that begins with a feasibility study and progresses through requirements definition, design, development, implementation and postimplementation. The series of steps or phases have defined goals and activities to perform with assigned responsibilities, expected outcomes and target completion dates.

Figure 3.11—Verification and Validation

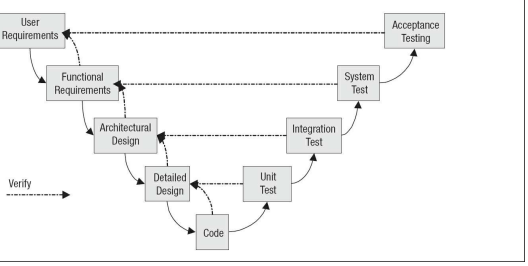


Figure 3.12—Traditional System Development Life Cycle Approach	
SDLC Phase	**General Description**
Phase 1—Feasibility Study	Determine the strategic benefits of implementing the system either in productivity gains or in future cost avoidance, identify and quantify the cost savings of a new system, and 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 (*purchased systems*)	Based on requirements defined, prepare a request for proposal outlining the entity requirements to invite bids from prospective 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—Postimplementation	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.

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. Another type of software development approach is the iterative approach where business requirements are developed and tested in iterations until the entire application is designed, built and tested. The traditional approach is useful in web applications in which prototypes of screens are necessary to aid in the completion of requirements and design.

As purchased packages have become more common, the design and development phases of the traditional life cycle are being replaced with the selection and configuration phases.

Note: The CISA candidate should be familiar with the SDLC phases and should be aware of what the IS auditor should look for when reviewing the feasibility study.

Some examples of measurements of critical success factors (CSFs) for an SDLC project could include productivity, quality, economic value and customer service, as shown in **figure 3.13**.

Figure 3.13—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

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:
- Unanticipated events. Because real projects rarely follow the sequential flow prescribed, iteration always occurs and creates 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

Moreover, 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.

3.5.2 DESCRIPTION OF TRADITIONAL SDLC PHASES

A traditional SDLC approach is made up of a number of distinct phases (six are shown in **figure 3.12**), each with a defined set of activities and outcomes. (There are other interpretations that use a slightly different number of phases with different names; for example, a seventh phase—maintenance—is often added.)

The following section describes in detail each phase's purpose and relationship to prior phases, the general activities performed and expected outcomes.

Phase 1—Feasibility Study
After the initial approval has been given to move forward with a project, an analysis begins to clearly define the need and to identify alternatives for addressing the need. This analysis is known as the 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, which 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. Intangible benefits such as improved morale may also be identified; however, benefits should be quantified wherever possible.

Within the feasibility study, the following typically are addressed:
- Define a time frame for the implementation of the required solution.
- Determine an optimum alternative risk-based solution for meeting business needs and general information resource requirements (e.g., whether to develop or acquire a system). Information resources include people, applications, technology, facilities and data organized and managed through a given

set of IT “natural” processes. Such processes can easily be mapped to SDLC and, with some more effort, to rapid application development (RAD).
- Determine whether an existing system can correct the situation with slight or no modification (e.g., workaround).
- Determine whether a vendor product offers a solution to the problem. Vendor products include services such as cloud infrastructure; platform and software as service; managed security services; and commercial off-the-shelf applications.
- Determine the approximate cost to develop the system to correct the situation.
- Determine whether the solution fits the business strategy.

Factors impacting whether to develop or acquire a system include:
- 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 some type of 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 and the information criteria the system should meet. The COBIT framework defines information criteria that should be incorporated in system requirements to address issues associated with effectiveness, efficiency, confidentiality, integrity, availability, compliance and reliability. The requirements definition phase also deals with overarching issues that are sometimes called nonfunctional requirements (e.g., access control). There is heightened focus in today’s marketplace about ensuring that security-related considerations are addressed early in the SDLC. Many IT security weaknesses can be corrected with a more critical focus on this topic within the context of the SDLC and in particular during the requirements definition.

To successfully complete the requirements definition phase, the project team should:
- 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.
- 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, nonautomated as well as automated, and how they wish to have them addressed by the system (e.g., access controls, regulatory requirements, management information needs and interface requirements).

The IS auditor should pay close attention to the degree the organization 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 need of data from creation/receipt, processing, storage, transmission and ultimately destruction.

From this interactive process, a general preliminary design of the system may be developed and presented to user management for their review, modification, approval and endorsement. 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 concerned 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.

IS auditors are involved at this stage to determine whether adequate security requirements have been defined to address, at a minimum, the confidentiality, integrity and availability requirements of the system. 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.

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 was 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 buy a vendor-supplied software package, the user must be actively involved in the package evaluation and selection process.

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. The vendor's software 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 a request for proposal (RFP) or invitation to tender (ITT). An RFP needs to be prepared separately for each case referred to 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 3.14**.

Figure 3.14—RFP Contents

Item	Description
Product vs. 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.

It is worth noting that a distinction can sometimes be drawn between ITT and RFP with respect to their applicability. When the product and related services are known in advance, a user organization will prefer an ITT so they can obtain the best combination of price and services. This is more applicable where 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, where the capability, experience and approach can be measured against the requirement. This is more applicable in system integration projects such as ERP and SCM that involve delivery or escrowing of source code.

Often, prior to the development of an RFP, organizations 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.

Note: The CISA candidate should be familiar with the RFP process.

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, etc.).

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

- Required hardware such as memory, disk space and server or client characteristics

- Operating system (OS) versions and patch levels supported
- Additional tools such as import and export tools
- Databases supported

Often 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 that adequate participation from various user groups is included when evaluating the product/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. With each short-listed vendor demonstrating their product following a scripted document, this also enables the project team to evaluate and finalize the product/vendor 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. The conference room pilot will enable the project team to understand the system with a hands-on session 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 can be cost-justified, an onsite visit can be even more beneficial. Whenever possible, the companies chosen should be those that use the products in a manner similar to the way the company will use the products. The 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 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 on with the vendor.

IS auditors are 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, the IS auditor should ensure that these controls are built into the software application.

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 and for defining 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:
- 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, they most often are 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.

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 taking into account 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 types of risk is scope creep—the process through which requirements change during development. Empirical studies have shown that a typical project can experience at least a 25 percent change in requirements throughout development, resulting in an increase in the effort and costs required for 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 would include, for example, 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 the 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 the IS auditor.

END OF DESIGN PHASE
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 INVOLVEMENT
The IS auditor involvement 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 e-commerce applications and other types of paperless environments). Additionally, the 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.

Phase 4A—Configuration
System configuration, as it relates to the SDLC, consists of defining, tracking and controlling changes in a purchased 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—Design to begin coding, moving the system one step closer to a final software product. 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

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 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 are:

- 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
- Hypertext Markup Language (HTML)
- 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 (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. These 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.

TESTING

Testing is an essential part of the 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 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. The IS auditor can play a preventive or detective role in the testing process.

ELEMENTS OF A SOFTWARE TESTING PROCESS

To guide the testing process and help ensure that all facets of the system function as expected, basic elements for application software testing activities have been defined and are discussed below.

Developed early in the life cycle and refined until the actual **testing phase**, test plans identify the specific portions of the system to be tested. Test plans 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. Often, cosmetic problems with the interface are classified as a lower severity and may not be fixed if time constraints become an issue for the project manager. This would be an example of a low severity, low priority defect. However, a defect in a report output such as spelling, look and feel, or specific content may also be a low-severity defect for the tester, but a high-priority defect for the business and thus would warrant resolution within the time constraints of the project. 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 two reciprocal approaches, to software testing:

- **Bottom up**—Begin testing of atomic units, such as programs or modules, and work upward until a complete system testing has taken place. The advantages are:
  - No need for stubs or drivers
  - Can be started before all programs are complete
  - Errors in critical modules are found early
- **Top down**—Follow the opposite path, either in depth-first or breadth-first search order. The advantages are:
  - Tests of major functions and processing are conducted early
  - Interface errors can be detected sooner
  - Confidence is raised in the system 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, etc.):

- **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.

TESTING CLASSIFICATIONS

The following tests relate, to varying degrees, to the above 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, where 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) that 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**—After the system staff is satisfied with their system tests, the new or modified system is ready for the acceptance testing, which 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 user acceptance testing (UAT) focusing on functional aspect 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.

Note: The CISA candidate should be familiar with the need for coding standards and with details on QA activities, software QA plan and the application QA function.

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 criteria 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 test the system in a manner as close to production as possible. For example, tests may be based around typical predefined, business process scenarios. If new business processes have been developed to accommodate the new or modified system, they 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 where both source 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 packaged 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 their 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 in which a subset of production data is used in a test environment, such production data may be altered to scramble the data so that the confidential nature of 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.

On completion of acceptance testing, the final step is usually a certification and accreditation process. Certification/accreditation should only be performed after the system is implemented and in operation for some time to produce the evidence needed for certification/accreditation processes. This process includes evaluating program documentation and testing effectiveness. The process will result 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.

Note: The CISA candidate should be familiar with details on independent certification, evaluating confidentiality for security certification and benchmarking.

When the tests are completed, the 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:
- **Alpha and beta testing**—An alpha version is an early version of the application system (or software product) submitted to internal users for testing. The alpha version may not contain all of the features that are planned for the final version. Typically, software goes through two stages of testing before it is 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 user acceptance testing, generally involves a limited number of external users. Beta testing is the last stage of testing, and normally involves real-world exposure, sending the beta version of the product to independent beta 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. Proofs of concept are early pilot tests—usually over interim platforms and with only basic functionalities.
- **White box testing**—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 and, therefore, 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. Applicable to integration (interface) and user acceptance testing processes.
- **Function/validation testing**—It is similar to system testing but is 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**—This is the process of feeding test data into two systems—the modified system and an alternative system (possibly the original system)—and comparing the results.
- **Sociability testing**—The purpose of these tests is to confirm that the new or modified system can operate in its target environment without adversely impacting existing systems. This should cover not only the platform that will perform primary application processing and interfaces with other systems but, in a client server or web development, changes to the desktop environment. Multiple applications may run on the user's desktop, potentially simultaneously, so it is important to test the impact of installing new dynamic link libraries (DLLs), making OS registry or configuration file modifications, and possibly extra memory utilization.

AUTOMATED APPLICATION TESTING

Automated testing techniques are used in this process. For example, 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.

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 effectiveness of the business application 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 (or dates) 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 commence 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.

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 naming 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:

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

Often the project sponsor's company 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.

One of the major challenges, therefore, 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.

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.

Step 1—Develop To-be Support Structures
Step 1 includes the development of:
- **Gap analysis:** One possible approach to determine the gap—the differences between the current support organization and the future one—is to schedule workshops with the appropriate staff members who are dealing with system operational tasks or support processes at present. This should also include representatives of the current help desk unit.
  - The gap analysis should be based on the results of those workshops and on the data that can be gathered from self-assessment where representatives of all present support units will take part. The topics of the gap analysis should include processes, skills, tools, headcount, services of the help desk and interfaces to other organizational units.
- **Role definitions:** The definitions for the required roles must be provided in detail.

Step 2—Establish Support Functions
Step 2 includes the development of service level agreements (SLAs) and implementation and training plans.

Service Level Agreement
The SLA should at least consider the following key attributes:
- Operating time
- Support time
- Mean time between failures (MTBF)
- Mean time to repair (MTTR)
- Technical support response time

All attributes of the SLA must be measurable with a reasonable effort.

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 Plans
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.

The training plans for the staff 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 takes into account 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 new organization structure.
- Review of items above.

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, the 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 needs to record student information in a database or spreadsheet, including student feedback for improving the training course.

DATA MIGRATION

A data conversion (also known as data porting) is required if the source and target systems utilize different field formats or sizes, file/database structures, or coding schemes. For example, a number may be stored as text, floating point or as binary-coded-decimal. Another example is the colors red, green and blue being represented using the codes "R," "G" and "B" on the existing system and "RD," "GN" and "BL" in the new system. Note that this conversion would also address a change in the field size. Conversions are often necessary when the source and target systems are on different hardware and/or OS platforms, and where different file or database structures (e.g., relational database, flat files, VSAM, etc.) are used. Another example of different coding schemes requiring conversion would be the use of different character representation schemes on the two systems (e.g., American standard code for information interchange [ASCII] vs. extended binary-coded decimal interchange code [EBCDIC]).

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, which 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. For example, 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 that will record any items that cannot be converted automatically
- Establishing responsibility for verifying and signing off on individual conversion steps and accepting the overall conversion (typically, the responsibility of the system owner)
- 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 test the conversion process end-to-end with real data
- Outsourcing the conversion process should be controlled 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 utilize 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 screenplay. This is a detailed listing of tasks for the production deployment of the new system. Within this plan decision points are defined to make "go" or "no-go" 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 supports 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 (e.g., EBCDIC–ASCII conversion), 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 data format of target system) in legacy database 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).

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:

1. The first consists of: (i) unload components to execute the unloading of the data from the new data structures, (ii) transfer components for the data conversion, and (iii) load components to execute the loading of the data into the legacy data structures.
2. The second consists of: (i) a log component to log the data modifications within the new data model during runtime within the service layer, (ii) transfer components for the data conversion, and (iii) load components to execute the load of the data into the legacy data structures.

The decision on which method to use for data conversion has to be made as part of the implementation project and should be based on the following criteria:

- Transaction volume
- Change degree of the data model

In summary, data migration projects begin with planning and preparation. The first step is to understand the new system's data structure. This is done 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.

The next step in data conversion is 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. The data conversion scripts are tested on a discrete selection of data that is carefully selected to include all cases. This 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 tested by executing tests 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. Data conversion can also be done manually if the conversion of data requires interpretation or if the conditions or data conversion rules are too complex for programming.

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

- Completeness of data conversion (i.e., the total number of records from the source database is transferred to the new database [assuming the number of fields is the same])
- Integrity of data (i.e., the data are not altered manually, mechanically or electronically by person, program, substitution or overwriting in the new system): Integrity problems also include errors due to transposition, transcription errors, and problems transferring particular records, fields, files and libraries.
- Storage and security of data under conversion (i.e., data are backed up before conversion for future reference or any emergency that might arise out of data conversion program management): Unauthorized copy or too many copies can lead to misuse, abuse or theft of data from the system.
- Consistency of data (i.e., the field/record called for from the new application should be consistent with that of the original application): This should enable consistency in repeatability of the testing exercise.
- Continuity (i.e., the new application should be able to continue with newer records as addition [append] and help in ensuring seamless business continuity)
- 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

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. See **figures 3.15, 3.16** and **3.17**.

Figure 3.15—Parallel Changeover

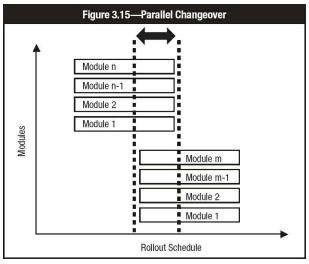


Figure 3.16—Phased Changeover

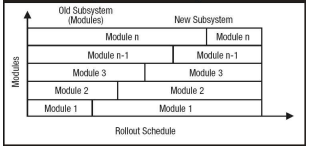
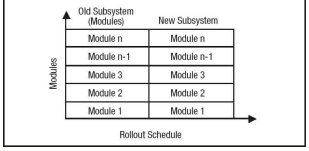


Figure 3.17—Abrupt Changeover



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 3.15** 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.

Some of the risk areas that may exist in the phased changeover include:
- 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

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.

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 areas 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 (duplicate or erroneous records may exist if data cleansing is not done correctly)

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 quality control 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.

Note: The CISA candidate should be familiar with the auditor's role in the certification process.

Phase 6—Postimplementation 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 postimplementation review should meet the following objectives:
- Assess the adequacy of the system.
  - Does the system meet user requirements and business objectives?
  - Have access controls been adequately defined and implemented?
- Evaluate the projected cost benefits or ROI measurements.
- Develop recommendations that address the system's inadequacies and deficiencies.
- Develop a plan for implementing the recommendations.

- Assess the development project process.
  - Were the chosen methodologies, standards and techniques followed?
  - Were appropriate project management techniques used?

It is important to note that, for a postimplementation 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 critical success factor measurements, see **figure 3.13**).

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 return on investment.

A postproject 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 postimplementation review has the objective of assessing and measuring the value the project has on the business (benefits realization).

Alternatively, an independent group not associated with the project implementation (internal or external audit) can perform a postimplementation review. The IS auditors performing this review should be independent of the system development process. Therefore, IS auditors involved in consulting with the project team on the development of the system should not perform this review. Unlike internal project team reviews, postimplementation reviews performed by IS auditors have a tendency 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 the 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 that control integrity still exists.

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

3.5.3 INTEGRATED RESOURCE MANAGEMENT SYSTEMS

A growing number of organizations—public and private—are shifting from separate groups of interrelated applications to a fully integrated corporate solution. Such solutions are often marketed as ERP solutions. Many vendors, mainly from Europe and the USA, have been focusing on this market and offer packages with commercial names such as SAP®, Oracle® Financials or SSG (Baan).

An integrated solutions implementation is a very large software acquisition project. The acquisition and implementation of an ERP system impacts the way the corporation does business, its entire control environment, technological direction and internal resources. Generally, a corporation 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 the corporation'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.

3.5.4 RISK ASSOCIATED WITH SOFTWARE DEVELOPMENT

There are many potential types of risk that can occur when designing and developing software systems.

One type of risk is business risk (or benefit risk), relating 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.

Another risk is project risk (or delivery risk), where 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, it may be completed late, if ever. There are many potential types of risk that can occur when designing and developing software systems. 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 over 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 foremost 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, organizations are 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.

The IS auditor should be aware that merely following an SDLC management approach does not ensure the successful completion of a development project. The 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 is performed in each project phase

(See section 3.12 Process Improvement Practices for more information.)

3.6 VIRTUALIZATION AND CLOUD COMPUTING ENVIRONMENTS

The need for greater scalability and agility in the market place and efforts to lower operating costs have driven the growth in both system virtualization and its use within cloud service providers (CSPs). To develop effective audit programs, the IS auditor must obtain a clear understanding of both virtualization and CSP architectures supporting the organization business applications and processes.

3.6.1 VIRTUALIZATION

As the sophistication and complexity of business processes grow, so does the supporting network and computing infrastructure. The concept of virtualization is generally believed to have its origins in the late 1960s and early 1970s when the mainframe-enabled robust time-sharing option (TSO) environments with several users accessing the OS concurrently, without impacting others, are also accessing the same OS—resulting in more effective use of central processing unit (CPU) processing, memory and storage space. However, the high cost of these resources at that time did not enable the ability to deploy these computing capabilities on the scale as seen in today's virtualization model. The same rationale for the TSO model is being applied today via virtualization because the majority of current server technologies are underutilized and capable of supporting much more than one or two server functions.

Data centers and many other organizations use virtualization techniques to create an abstraction of the physical hardware and make large pools of logical resources consisting of CPUs, memory, disks, file storage, applications and networking. This approach enables greater availability of these resources to the user base. The main focus of virtualization is to enable a single physical computing environment to run multiple logical, yet independent, systems at the same time.

The most common use for full virtualization is operational efficiency, which uses existing hardware more efficiently by placing greater loads on each computer. Second, using full virtualization of desktops enables end users to have one computer hosting multiple OSs if needed to support various OS-dependent applications. Furthermore, the IT organization can better control deployed OSs to ensure that they meet the organization's security requirements, security threat and respective control requirements are dynamic, and the virtual desktop images can be changed to respond to new threats.

Elements of the virtualized computing environment are normally comprised of the following:
- Server or other hardware product
- Virtualization hypervisor: A piece of computer software, firmware or hardware that creates and runs virtual machine environment—normally called the "host."
- Guest machine: Virtual environment elements (e.g., OS, switches, routers, firewalls, etc.) residing on the computer on which a hypervisor host machine has been installed

There are two methods of deploying a fully virtualized environment. **Figure 3.18** compares these architectures.
- **Bare metal/native virtualization** occurs when the hypervisor runs directly on the underlying hardware, without a host OS.
- **Hosted virtualization** occurs when the hypervisor runs on top of the host OS (Windows, Linux or MacOS). The hosted virtualization architectures usually have an additional layer of software (the virtualization application) running in the guest OS that provides utilities to control the virtualization while in the guest OS, such as the ability to share files with the host OS.

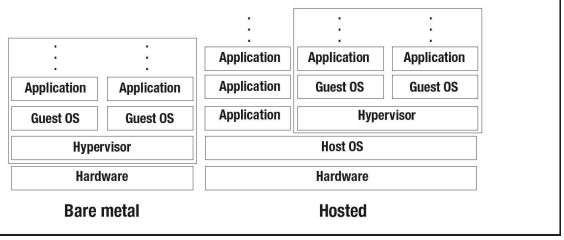
Key Risk Areas

Overall, migrating computing resources to a virtualized environment does not change the threat plane for most of the systems' vulnerabilities and threats. If a service has inherent vulnerabilities on a physical server or network product and it is migrated to a virtualized server, the service remains vulnerable to exploitation. However, the use of virtualization may also provide additional virtual environment attack vectors (e.g., hypervisor misconfiguration or security flaws, memory leakage, etc.), thus increasing the likelihood of successful attacks. The following types of high-level risk are representative of the majority of virtualized systems in use:
- Rootkits on the host installing themselves as a hypervisor below the OS, enabling the interception of any operations of the guest OS (i.e., logging password entry, etc.): Antivirus software may not detect this, because the malware runs below the entire OS.
- Default and/or improper configuration of the hypervisor partitioning resources (CPU, memory, disk space and storage): This can lead to unauthorized access to resources, one guest OS injecting malware into another or placing malware code into another guest OS's memory.
- On hosted virtualization, mechanisms called guest tools enable a guest OS to access files, directories, the copy/paste buffer, and other resources on the host OS or another guest OS: This functionality can inadvertently provide an attack vector for malware or allow an attacker to gain access to particular resources.
- Snapshots/images of guests' environments contain sensitive data (such as passwords, personal data, etc.) like a physical hard drive: These snapshots pose a greater risk than images because snapshots contain the contents of random access memory (RAM) at the time that the snapshot was taken, and this might include sensitive information that was not stored on the drive itself.

- In contrast to bare metal installations, hosted virtualization products rarely have hypervisor access controls: Therefore, anyone who can launch an application on the host OS can run the hypervisor. The only access control is whether someone can log into the host OS.

Typical Controls
The IS auditor will need to understand the following concepts:
- Hypervisors and guest images (OS and networks) are securely configured according to industry standards. Apply hardening to these virtual components as closely as one would to a physical server, switch, router, firewall or other computing device.
- Hypervisor management communications should be protected on a dedicated management network. Management communications carried on untrusted networks should be encrypted, and encryption should encapsulate the management traffic.
- The hypervisor should be patched as the vendor releases the fixes.
- The virtualized infrastructure should be synchronized to a trusted authoritative timeserver.
- Unused physical hardware should be disconnected from the host system.
- All hypervisor services, such as clipboard- or file-sharing between the guest OS and the host OS, should be disabled unless they are needed.
- Host inspection capabilities should be enabled to monitor the security of each guest OS. Hypervisor security services can allow security monitoring even when the guest OS is compromised.
- Host inspection capabilities should be enabled to monitor the security of activity occurring between guest OSs. Of special focus is communications in a non-virtualized environment carried and monitored over networks by network security controls (such as network firewalls, security appliances, and network IDPS sensors).

Figure 3.18—Full Virtualization Architectures



- File integrity monitoring of the hypervisor should be used to monitor for signs of compromise.

See sections 5.4.1 LAN Security and 5.13 Cloud Computing for details regarding the risk and controls over virtual computing environments.

3.7 BUSINESS APPLICATION SYSTEMS

To develop effective audit programs, the IS auditor must obtain a clear understanding of the application system under review. Some types of application systems and the related processes are described in the following sections.

Numerous financial and operational functions are computerized for the purpose of improving efficiency and increasing the reliability of information. These applications range from traditional (including general ledger, accounts payable and payroll) to industry-specific (such as bank loans, trade clearing and material requirements planning). Given their unique characteristics, computerized application systems add complexity to audit efforts. These characteristics may include limited audit trails, instantaneous updating and information overload. Application systems may reside in the various environments that follow.

3.7.1 E-COMMERCE

E-commerce is the buying and selling of goods online, usually via the Internet. Typically, a web site will advertise goods and services, and the buyer will fill in a form on the web site to select the items to be purchased and provide delivery and payment details or banking services such as transfers and payment orders. The web site may gather details about customers and offer other items that may be of interest. The cost of a brick-and-mortar store is avoided, and the savings are often a benefit to the customers, sometimes leading to spectacular growth. The term e-business includes buying and selling online as well as other aspects of online business such as customer support or relationships between businesses.

E-commerce, as a general model, uses technology to enhance the processes of commercial transactions among a company, its customers and business partners. The used technology can include the Internet, multimedia, web browsers, proprietary networks, automatic teller machines (ATMs) and home banking, and the traditional approach to electronic data interchange (EDI). However, the primary area of growth in e-commerce is through the use of the Internet as an enabling technology.

E-commerce Models
E-commerce models include the following:
- **Business-to-consumer (B-to-C) relationships**—The greatest potential power of e-commerce comes from its ability to redefine the relationship with customers in creating a new convenient, low-cost channel to transact business. Companies can tailor their marketing strategies to an individual customer's needs and wants. As more of its business shifts online, a company will have an enhanced ability to track how its customers interact with it.
- **Business-to-business (B-to-B) relationships**—The relationship among the selling services of two or more businesses opens up the possibility of reengineering business processes across the boundaries that have traditionally separated external entities from each other. Because of the ease of access and the ubiquity of the Internet; for example, companies can build business processes (order processing, payments and after-sale service) that combine previously separated activities. The result is a faster, higher-quality and lower-cost set of transactions. The market has even created a subdivision of B-to-B called business-to-small business (B-to-SB) relationships.
- **Business-to-employee (B-to-E) relationships**—Web technologies also assist in the dissemination of information to and among an organization's employees.
- **Business-to-government (B-to-G) relationships**—Covers all transactions between companies and government organizations. Currently this category is expanding quite rapidly as governments use their own operations to promote awareness and growth of e-commerce. In addition to public procurement,

administrations may also offer the option of electronic interchange for such transactions as VAT returns and the payment of corporate taxes.

E-COMMERCE ARCHITECTURES

There are a large number of choices to be made in determining an appropriate e-commerce architecture. Initially, e-commerce architectures were either two-tiered (i.e., client browser and web server) or three-tiered (i.e., client browser, web server and database server). With increasing emphasis on integrating the web channel with a business' internal legacy systems and the systems of its business partners, company systems now typically will run on different platforms, running different software and with different databases. It is also the case that, in addition to supporting browser connections, companies eventually may move in the direction of supporting connections from active content clients, mobile phones or other wireless devices, and host-to-host connections. Systems that rely on multiple computer platforms are described as "*n*-tiered" also referred to as multitiered architectures. The *n*-tiered architecture separates presentation, application processing and data management functions, thereby allowing modification to be made to any of these functions instead of reworking all as one application. Types of architectures include:

- Single-tier architecture is a client-based application running on a single computer.
- Two-tier architecture is composed of the client and server.
- Three-tier architecture is comprised of the following:
  - The presentation tier displays information which users can access directly such as a web page, or an OS's GUI.
  - The application tier (business logic/applications) controls an application's functionality by performing detailed processing.
  - The data tier is usually comprised of the database servers, file shares, etc.) and the data access layer that encapsulates the persistence mechanisms and exposes the data.

The challenge of integrating diverse technologies within and beyond the business has increasingly led companies to move to component-based systems that utilize a middleware infrastructure based around an application server. This supports current trends in the evolution of software development: build systems from proven quality and cataloged components, just as hardware is built. While this is yet to be fully realized, component models—notably Microsoft Component Object Model (COM) and Oracle Enterprise JavaBeans (EJB)—are widely used and fall under the grouping of "mobile code." Mobile code is software transferred between systems (i.e., transferred across a network) and executed on a local system using cross-platform code without explicit installation by the recipient computer (e.g., Adobe® Flash®, Shockwave®, Java applets, VBScripts, ActiveX, etc.). The continued adoption of mobile code brings another vector for the spread of malware via ever-evolving delivery vectors ranging from email, malicious web sites and mobile device applications.

E-components often seen in a B-to-C system include marketing, sales and customer service components (e.g., personalization, membership, product catalog, customer ordering, invoicing, shipping, inventory replacement, online training and problem notification).

Application servers support a particular component model and provide services (such as data management, security and transaction management) either directly or through connection to another service or middleware product such as MQSeries. A number of major software vendors offer application servers.

Application servers in conjunction with other middleware products provide for multitiered systems (i.e., a business transaction can span multiple platforms and software layers). For example, a system's presentation layer typically will consist of a browser or other client application. A web server will be used to manage web content and connections; business logic and other services will be provided by the application server; and one or more database(s) will be used for data storage.

Databases play a key role in most e-commerce systems, maintaining data for web site pages, accumulating customer information and storing click-stream data for analyzing web site usage. To provide full functionality and achieve back-end efficiencies, an e-commerce system may involve connections to in-house legacy systems—accounting, inventory management or an ERP system—or business partner systems. Thus, further business logic and data persistence tiers are added.

For security reasons, persistent customer data should not be stored on web servers that are exposed directly to the Internet. Extensible Markup Language (XML) is also likely to form an important part of an organization's overall e-commerce architecture. While originally conceived as a technique to facilitate electronic publishing, XML was quickly seized on as a medium that could store and enclose any kind of structured information so it could be passed between different computing systems. XML has emerged as a key means of exchanging a wide variety of data on the web and elsewhere. In addition to basic XML, a variety of associated standards has been and is continuing to be developed. Some of these include:

- **Extensible Stylesheet Language (XSL)**—Defines how an XML document is to be presented (e.g., on a web page)
- **XML query (XQuery)**—Deals with querying XML format data
- **XML encryption**—Deals with encrypting, decrypting and digitally signing XML documents

A particularly important offshoot of XML is web services. Web services represent a way of using XML format information to remotely invoke processing. Because a web services message can contain both an XML document and a corresponding schema defining the document, in theory, it is self-describing and assists in achieving the goal of "loose coupling." If the format of a web services message changes, the receiving web services will still work, provided the accompanying schema is updated. This advantage, combined with the support for web services that has emerged from major software industry players such as IBM and Microsoft, means web services is now the key middleware to connect distributed web systems.

Cooperation between the different software vendors means that web services can interoperate, irrespective of hardware, OS and programming language.

It is necessary to reach some agreement on metadata definitions for web services to serve as a means of enabling cooperative processing across organizational boundaries. (Metadata are data about data, and the term is referred to in web services' standards as ontology). Web services may be successfully called and the resulting XML data may be successfully parsed by the calling program, but to use these data effectively it is necessary to understand the business meaning of the data. This is similar to previous attempts at interorganizational computing (such as EDI), where it was necessary to agree in advance on electronic document formats and meanings. The Object Management Group (OMG) and *ISO/IEC 19510:2013: Information technology —Object Management Group Business Process Model and Notation* have established a recognized industry-standard XML documents and standard business process definitions that can be represented in XML.

E-COMMERCE RISK

E-commerce, as any other form of commerce, depends on the existence of a level of trust between two parties. For example, the Internet presents a challenge between the buyer and seller, similar to those a catalog or direct-mail retailer faces. The challenges are proving to the buyer that the seller is who they say they are, proving to the buyer that their personal information, such as credit card numbers (and other personally identifiable information), remains confidential and that the seller cannot later refute the occurrence of a valid transaction. Some of the most important elements at risk are:

- **Confidentiality**—Potential consumers are concerned about providing unknown vendors with personal (sometimes sensitive) information for a number of reasons including the possible theft of credit card information from the vendor following a purchase. Connecting to the Internet via a browser requires running software on the computer that has been developed by someone unknown to the organization. Moreover, the medium of the Internet is a broadcast network, which means that whatever is placed on it is routed over wide-ranging and essentially uncontrolled paths. The current trend of outsourcing and hosting services on the cloud expands the risk perimeter beyond the boundaries of the transacting entity.
- **Integrity**—Data, both in transit and in storage, could be susceptible to unauthorized alteration or deletion (i.e., hacking or the e-business system itself could have design or configuration problems).
- **Availability**—The Internet holds out the promise of doing business on a 24-hour, seven-day-a-week basis. Hence, high availability is important, with any system's failure becoming immediately apparent to customers or business partners.
- **Authentication and nonrepudiation**—The parties to an electronic transaction should be in a known and trusted business relationship, which requires that they prove their respective identities before executing the transaction in preventing man-in-the-middle attacks (i.e., preventing the seller from being an impostor). Then, after the fact, there must be some manner of ensuring that the transacting parties cannot deny that the transaction was entered into and the terms on which it was completed.
- **Power shift to customers**—The Internet gives consumers unparalleled access to market information and generally makes it easier to shift between suppliers. Firms participating in e-business need to make their offerings attractive and seamless in terms of service delivery. This will involve not only system design, but also reengineering of business processes. Back-end support processes need to be as efficient as possible because, in many cases, doing business over the Internet forces down prices (e.g., online share brokering). To avoid losing their competitive advantage of doing business online, firms need to enhance their services, differentiate from the competition and build additional value. Hence, the drive to personalize web sites by targeting content based on analyzed customer behavior and allowing direct contact with staff through instant messaging technology and other means.

It is important to take into consideration the importance of security issues that extend beyond confidentiality objectives.

E-COMMERCE REQUIREMENTS

Some e-commerce requirements include:
- Build a business case (IT as an enabler).
- Develop a clear business purpose.
- Use technology to first improve costs.
- Build business case around the four C's: customers, costs, competitors and capabilities.

Other requirements for e-commerce include the following:
- **Top-level commitment**—Because of the breadth of changes required (i.e., business processes, company culture, technology and customer boundaries), e-commerce cannot succeed without a clear vision and strong commitment from the top of the organization.
- **Business process reconfiguration**—Technology is not the key innovation needed to make e-commerce work, but it is the ingenuity needed to envision how that technology can enable the company to fundamentally reconfigure some of its basic business processes. This requires thinking that is outside-the-box and outside-the-walls (i.e., looking outside of the organization and understanding what customers are doing and how changes in the overall process can create new value for them).
- **Links to legacy systems**—Organizations must take seriously the requirement to accelerate response times, provide real interaction to customers and customize responses to individual customers. Specifically, in applying enterprise application integration (EAI), organizations must create online interfaces and make sure those interfaces communicate with existing databases and systems for customer service and order processing. A term often referred to in establishing this communication is middleware, which is defined as independent software and services that distributed business applications use to share computing resources across heterogeneous technologies. A range of middleware technologies—message brokers, gateways, process managers, data transformation software and file transfer—are likely to be deployed to create an integration infrastructure. Increasingly, integration will be viewed not as a responsibility of an individual application development team, but as something to be managed across the organization using a standard approach and technologies.

E-COMMERCE AUDIT AND CONTROL ISSUES (GOOD PRACTICES)

When reviewing the adequacy of contracts in e-commerce applications, audit and control professionals should assess applicable use of the following items:
- For B-to-C, B-to-B, B-to-E and B-to-G, interconnection agreements need to be established and documented prior to engaging in the e-commerce agreement. This can be as simple as accepting terms of use for B-to-C and B-to-E systems to detailed terms and conditions to be in place before the e-commerce interconnections can be established.
- Security mechanisms and procedures that, taken together, constitute a security architecture for e-commerce (e.g., Internet firewalls, public key infrastructure [PKI], encryption, certificates, PCI DSS compliance and password management)
- Firewall mechanisms that are in place to mediate between the public network (the Internet) and an organization's private network
- A process whereby participants in an e-commerce transaction can be identified uniquely and positively (e.g., process of using some combination of public and private key encryption and certifying key pairs)
- Digital signatures so the initiator of an e-commerce transaction can be uniquely associated with it. Attributes of digital signatures include:
  - The digital signature is unique to the person using it.
  - The signature can be verified.
  - The mechanism for generating and affixing the signature is under the sole control of the person using it.
  - The signature is linked to data in such a manner that if the data are changed, the digital signature is invalidated.
- Infrastructure to manage and control public key pairs and their corresponding certificates which include:
  - Certificate authority (CA)—Attests, as trusted provider of the public/private key pairs, to the authenticity of the owner (entity or individual) to whom a public/private key pair has been given. The process involves a CA who makes a decision to issue a certificate based on evidence or knowledge obtained in verifying the identity of the recipient. On verifying the identity of the recipient, the CA signs the certificate with its private key for distribution to the user. On receipt, the user will decrypt the certificate with the CA's public key (e.g., commercial CAs such as Verisign® provide public keys on web browsers). The ideal CA is authoritative (someone that the user trusts) for the name or key space it represents.
  - Registration authority (RA)—An optional entity separate from a CA that would be used by a CA with a very large customer base. CAs use RAs to delegate some of the administrative functions associated with recording or verifying some or all of the information needed by a CA to issue certificates or certification revocation lists (CRLs) and to perform other certificate management functions. However, with this arrangement, the CA retains sole responsibility for signing either digital certificates or CRLs. If an RA is not present in the established PKI structure, the CA is assumed to have the same set of capabilities as defined for an RA.
  - Certification revocation list (CRL)—Instrument for checking the continued validity of the certificates. If a certificate is compromised, if the holder is

no longer authorized to use the certificate or if there is a fault in binding the certificate to the holder, the certificate must be revoked and taken out of circulation as rapidly as possible and all parties in the trust relationship must be informed. The CRL is usually a highly controlled online database through which subscribers and administrators may determine the status of a target partner's certificate.
  - Certification practice statement (CPS)—A detailed set of rules governing the certificate authority's operations. The CPS provides an understanding of the value and trustworthiness of certificates issued by a given CA, the terms of the controls that an organization observes, the method used to validate the authenticity of certificate applicants and the CA's expectations of how its certificates may be used.
- Procedures in place to control changes to an e-commerce presence
- E-commerce application logs, which are monitored by responsible personnel. This includes OS logs and console messages, network management messages, firewall logs and alerts, router management messages, intrusion detection alarms, application and server statistics, and system integrity checks.
- Methods and procedures to recognize security breaches when they occur (network and host-based intrusion detection systems [IDSs])
- Features in e-commerce applications to reconstruct the activity performed by the application
- Protection in place to ensure that data collected about individuals are not disclosed without the individuals' consent nor used for purposes other than that for which they are collected
- Means to ensure confidentiality of data communicated between customers and vendors (safeguarding resources such as through encrypted Secure Sockets Layer [SSL])
- Mechanisms to protect the presence of e-commerce and supporting private networks from computer viruses and to prevent them from propagating viruses to customers and vendors
- Features within the e-commerce architecture to keep all components from failing and allow them to repair themselves, if they should fail
- Plan and procedure to continue e-commerce activities in the event of an extended outage of required resources for normal processing
- Commonly understood set of practices and procedures to define management's intentions for the security of e-commerce
- Shared responsibility within an organization for e-commerce security
- Communications from vendors to customers about the level of security in an e-commerce architecture
- Regular program of audit and assessment of the security of e-commerce environments and applications to provide assurance that controls are present and effective

3.7.2 ELECTRONIC DATA INTERCHANGE

EDI replaces the traditional paper document exchange, such as medical claims and records, purchase orders, invoices, or material release schedules. Therefore, the proper controls and edits need to be built within each company's application system to allow this communication to take place.

General Requirements

An EDI system requires communications software, translation software and access to standards. Communications software moves data from one point to another, flags the start and end of an EDI transmission and determines how acknowledgments are transmitted and reconciled. Translation software helps build a map and shows how the data fields from the application correspond to elements of an EDI standard. Later, it uses this map to convert data back and forth between the application and EDI formats.

To build a map, an EDI standard appropriate for the kind of EDI data to be transmitted is selected. For example, there are specific standards for medical claims, patient records, invoices, purchase orders, advance shipping notices, etc.

The final step is to write a partner profile that tells the system where to send each transaction and how to handle errors and exceptions.

In summary, components of an EDI process include system software and application systems. EDI system software includes transmission, translation and storage of transactions initiated by or destined for application processing. EDI is also an application system in that the functions it performs are based on business needs and activities. The applications, transactions and trading partners supported will change over time, and the intermixing of transactions, purchase orders, shipping notices, invoices and payments in the EDI process makes it necessary to include application processing procedures and controls in the EDI process.

In reviewing EDI, IS auditors need to be aware of the two approaches related to EDI: the traditional proprietary version of EDI used by large companies and government parties, and the development of EDI through the publicly available commercial infrastructure offered through the Internet. The difference between the approaches relates to cost, where use of a public commercial infrastructure such as the Internet provides significantly reduced costs versus development of a customized proprietary approach. From a security standpoint, risk associated with not having a completely trustworthy relationship arise in addressing Internet security and risk.

Traditional EDI

Moving data in a batch transmission process through the traditional EDI process generally involves three functions within each trading partner's computer system:

1. **Communications handler**—Process for transmitting and receiving electronic documents between trading partners via dial-up lines, public-switched network, multiple dedicated lines or a value-added network (VAN). VANs use computerized message switching and storage capabilities to provide electronic mailbox services similar to a post office. The VAN receives all the outbound transactions from an organization, sorts them by destination and passes them to recipients when they log on to check their mailbox and receive transmissions. VANs may also perform translation and verification services. VANs specializing in EDI applications also provide technical support, help desk and troubleshooting assistance for EDI and telecommunications problems. VANs help in configuration of software, offer upgrades to telecommunications connectivity, provide data and computer security, audit and trace transactions, recover lost data, and confirm service reliability and availability.
2. **EDI interface**—Interface function that manipulates and routes data between the application system and the communications handler. The interface consists of two components:
   - EDI translator—This device translates the data between the standard format (ANSI X12) and a trading partner's proprietary format.
   - Application interface—This interface moves electronic transactions to or from the application systems and performs data mapping. Data mapping is the process by which data are extracted from the EDI translation process and integrated with the data or processes of the receiving company. The EDI interface may generate and send functional acknowledgments, verify the identity of partners and check the validity of transactions by checking transmission information against a trading partner master file. Functional acknowledgments are standard EDI transactions that tell the trading partners that their electronic documents were received. Different types of functional acknowledgments provide various levels of detail and can, therefore, act as an audit trail for EDI transactions.
3. **Application system**—The programs that process the data sent to, or received from, the trading partner. Although new controls should be developed for

the EDI interface, the controls for existing applications, if left unchanged, are usually unaffected.

Application-initiated transactions (such as purchase orders from the purchasing system) are passed to a common application interface for storage and interpretation. All outbound transactions are formatted according to an externally defined standard and batched by destination and transaction type by the translator. The batches of transactions, like functional groups, are routed to the communications processor for transmission. This entire process is reversed for inbound transactions, including invoices destined for the purchasing and accounts payable systems. Controls need to recognize and deal with error conditions and provide feedback on the process for the EDI system to be considered well managed.

Web-based EDI

Web-based EDI has come into prominence because of the following:

- Internet-through-Internet service providers offer a generic network access (i.e., not specific to EDI) for all computers connected to the Internet, whereas VAN services have typically used a proprietary network or a network gateway linked with a specific set of proprietary networks. The result is a substantially reduced cost to EDI applications.
- Its ability to attract new partners via web-based sites to exchange information, take orders and link the web site to back-end order processing and financial systems via EDI
- New security products available to address issues of confidentiality, authentication, data integrity and nonrepudiation of origin and return
- Improvements in the X12 EDI formatting standard

Web-based EDI trading techniques aim to improve the interchange of information between trading partners, suppliers and customers by bringing down the boundaries that restrict how they interact and do business with each other. For example, the use of Internet service provider (ISP)-related services can provide functions similar to those of the more traditional VANs, but with a much broader array of available services (i.e., processing transactions of all types through the Internet). This is beneficial particularly for smaller organizations wanting to enter the e-commerce EDI market because ISPs have a ready network infrastructure of servers offering email, web services and the network of routers, and modems attached to a permanent, high-speed Internet “backbone” connection.

3.7.3 EDI RISK AND CONTROLS

The hybrid nature of EDI adds a new dimension to the design and auditing of the EDI process. The traditional procedures for managed and controlled implementation of system software—such as requirements definition, version and release identification, testing and limited implementation with a fallback strategy—apply to software used for EDI. In addition, there are issues and risk unique to EDI.

Transaction authorization is the biggest EDI risk. Because the interaction between parties is electronic, there is no inherent authentication occurring. Computerized data can look the same no matter what the source and do not include any distinguishing human element or signature.

Where responsibilities of trading partners are not clearly defined by a trading partner agreement, there could be uncertainty related to specific, legal liability. Therefore, it is important that, to protect both parties, any agreement is codified legally in what is known as a trading partner agreement. Another risk is the loss of business continuity. Corruption of EDI applications, whether done innocently or deliberately, could affect every EDI transaction undertaken by a company. This would have a negative impact on both customer and vendor relations. In an extreme situation, it could ultimately affect the ability of a company to stay in business.

Additional security types of risk include:

- Unauthorized access to electronic transactions
- Deletion or manipulation of transactions prior to or after establishment of application controls
- Loss or duplication of EDI transmissions
- Loss of confidentiality and improper distribution of EDI transactions while in the possession of third parties

3.7.4 CONTROLS IN THE EDI ENVIRONMENT

Security risk can be addressed by enforcing general controls and establishing an added layer of application control procedures over the EDI process that can take over where traditional application controls leave off. These controls need to secure the current EDI activity as well as historical activities that may be called on to substantiate business transactions should a dispute arise.

To protect EDI transmissions, the EDI process should include the following electronic measures:

- Standards should be set to indicate that the message format and content are valid to avoid transmission errors.
- Controls should be in place to ensure that standard transmissions are properly converted for the application software by the translation application.
- The receiving organization must have controls in place to test the reasonableness of messages received. This should be based on a trading partner’s transaction history or documentation received that substantiates special situations.
- Controls should be established to guard against manipulation of data in active transactions, files and archives. Attempts to change records should be recorded by the system for management review and attention.
- Procedures should be established to determine messages are only from authorized parties and transmissions are properly authorized.
- Direct or dedicated transmission channels among the parties should exist to reduce the risk of tapping into the transmission lines.
- Data should be encrypted using algorithms agreed on by the parties involved.
- Electronic signatures should be in the transmissions to identify the source and destination.
- Message authentication codes should exist to ensure that what is sent is received.

The EDI process needs the ability to detect and deal with transactions that do not conform to the standard format or are from/to unauthorized parties. Options for handling detected errors include requesting retransmissions or manually changing the data.

The critical nature of many EDI transactions, such as orders and payments, requires that there be positive assurances that the transmissions were complete. The transactions need to be successfully passed from the originating computer application to the destination organization. Methods for providing these assurances include internal batch total checking, run-to-run and transmission record count balancing, and use of special acknowledgment transactions for functional acknowledgments.

Organizations desiring to exchange transactions using EDI are establishing a new business relationship. This business relationship needs to be defined so

both parties can conduct business in a consistent and trusting manner. This relationship usually is defined in a legal document called a trading partner agreement. The document should define the transactions to be used, responsibilities of both parties in handling and processing the transactions, as well as the written business terms and conditions associated with the transactions.

The evolving nature of EDI means that the transaction standards are also evolving, particularly with use of the Internet as an available, low-cost primary business transaction delivery system. Specifically, not all trading partners desire or need to use the current standard. As a result, the EDI process needs to adapt to changes in standards and be able to support multiple versions of the standard as ISPs offer more EDI-related services (vs. VANs).

Other issues relate to many organizations with a large, installed base of applications that need to be retrofit to accommodate EDI. In addition, not all transactions will be processed through EDI. Some transactions will continue to be processed in the traditional way. The application processing control procedures must be modified to include the EDI transaction processing and the dual sources/destinations.

Receipt of Inbound Transactions
Controls should ensure that all inbound EDI transactions are accurately and completely received (communication phase), translated (translation phase) and passed to an application (application interface phase) as well as processed only once.

The control considerations for receipt of inbound transactions are as follows:
- Use appropriate encryption techniques when using public Internet infrastructures for communication in assuring confidentiality, authenticity and integrity of transactions.
- Perform edit checks to identify erroneous, unusual or invalid transactions prior to updating an application.
- Perform additional computerized checking to assess transaction reasonableness, validity, etc. (Consider expert system front ends for complex comparisons.)
- Log each inbound transaction on receipt.
- Use control totals on receipt of transactions to verify the number and value of transactions to be passed to each application; reconcile totals between applications and with trading partners.
- Segment count totals built into the transaction set trailer by the sender.
- Control techniques in the processing of individual transactions such as check digits on control fields, loop or repeat counts.
- Ensure the exchange of control totals of transactions sent and received between trading partners at predefined intervals.
- Maintain a record of the number of messages received/sent and validate these with the trading partners periodically.
- Arrange for security over temporary files and data transfer to ensure that inbound transactions are not altered or erased between time of transaction receipt and application updates.

Outbound Transactions
Controls should ensure that only properly authorized outbound transactions are processed. This includes the objectives that outbound EDI messages are initiated on authorization, that they contain only preapproved transaction types and that they are sent only to valid trading partners.

The control considerations for outbound transactions are as follows:
- Control the set up and change of trading partner details.
- Compare transactions with trading partner transaction profiles.
- Match the trading partner number to the trading master file (prior to transmission).
- Limit the authority of users within the organization to initiate specific EDI transactions.
- Segregate initiation and transmission responsibilities for high-risk transactions.
- Document management sign-off on programmed procedures and subsequent changes.
- Log all payment transactions to a separate file, which is reviewed for authorization before transmission.
- Segregate duties within the transaction cycle, particularly where transactions are automatically generated by the system.
- Segregate access to different authorization processes in a transaction cycle.
- Report large (value) or unusual transactions for review prior to or after transmission.
- Log outbound transactions in a secure temporary file until authorized and due for transmission.
- Require paperless authorization that would establish special access to authorization fields (probably two levels, requiring the intervention of different users) within the computer system.

Auditing EDI
The IS auditor must evaluate EDI to ensure that all inbound EDI transactions are received and translated accurately, passed to an application, and processed only once.

To accomplish this goal, IS auditors must review the following:
- Internet encryption processes put in place to ensure authenticity, integrity, confidentiality and nonrepudiation of transactions
- Edit checks to identify erroneous, unusual or invalid transactions prior to updating the application
- Additional computerized checking to assess transaction reasonableness and validity
- Each inbound transaction to ensure that it is logged on receipt
- The use of control totals on receipt of transactions to verify the number and value of transactions to be passed to each application and reconcile totals between applications and with trading partners
- Segment count totals built into transaction set trailers by the sender
- Transaction set count totals built into the functional group headers by the sender
- Batch control totals built into the functional group headers by the sender
- The validity of the sender against trading partner details by:
  - Using control fields within an EDI message at either the transaction, function, group or interchange level (often within the EDI header, trailer or control record)
  - Using VAN sequential control numbers or reports (if applicable)
  - Sending an acknowledgment transaction to inform the sender of message receipt. The sender should then match this against a file/log of EDI messages sent.

EDI audits also involve:

- **Audit monitors**—Devices can be installed at EDI workstations to capture transactions as they are received. Such transactions can be stored in a protected file for use by the auditor. Consideration should be given to storage requirements for voluminous amounts of data.
- **Expert systems**—Within the context of utilizing the computer system for internal control checks, consideration should be given to having audit monitors evaluate the transactions received. Based upon judgmental rules, the system can determine the audit significance of such transactions and provide a report for the auditor's use.

As use of EDI becomes more widespread, additional methods for auditing transactions will be developed. It is important to stay current with these developments.

3.7.5 EMAIL

Email may be the most heavily used feature of the Internet or LANs in an organization. At the most basic level, the email process can be divided into two principal components:
- **Mail servers**—Hosts that deliver, forward and store mail
- **Clients**—Interface with users and allow users to read, compose, send and store email messages

Email messages are sent in the same way as most Internet data. When a user sends an email message, it is first broken up by the Transmission Control Protocol (TCP) into Internet Protocol (IP) packets. Those packets are then sent to an internal router (a router that is inside the user's network) that examines the address. Based on the address, the router decides whether the mail is to be delivered to someone on the same network or to someone outside of the network. If the mail goes to someone on the same network, the mail is delivered to them. If the mail is addressed to someone outside the network, it may pass through a firewall, which is a security mechanism (hardware/software) that shields the network from the broader Internet so intruders cannot break into the network. The firewall keeps track of messages and data going into and out of the network—to and from the Internet. The firewall also can prevent certain packets from getting through it.

Once out on the Internet, the message is sent to an Internet router. The router examines the address, determines where the message should be sent and sends the message on its way. A gateway at the receiving network receives the email message. This gateway uses TCP to reconstruct the IP packets into a full message. The gateway then translates the message into the protocol the target network uses and sends it on its way. The message may be required to also pass through a firewall on the receiving network. The receiving network examines the email address and sends the message to a specific mailbox.

A user can also attach binary files such as pictures, videos, sounds and executable files to the email message. To do this, the user must encode the file in a way that will allow it to be sent across the network. The receiver will have to decode the file once it is received. There are a variety of different encoding schemes that can be used. Some email software packages automatically do the encoding for the user and the decoding on the receiving end.

When a user sends email to someone on the Internet or within a closed network, that message often has to travel through a series of networks before it reaches the recipient. These networks might use different email formats. Gateways perform the job of translating email formats from one network to another so the messages can make their way through all the networks. An email message is made up of binary data, usually in the ASCII text format. ASCII is a standard that allows any computer, regardless of its OS or hardware, to read the text. ASCII code describes the characters that users see on their computer screens.

There are several email protocols in use. The following protocols are the primary protocols the IS auditor will need to be familiar with during the review of email services:
- Outgoing email
  - Simple Mail Transport Protocol (SMTP): can only be used to send emails, not to receive them
- Incoming email
  - Post Office Protocol (POP)
  - Internet Message Access Protocol (IMAP)
  - Hypertext Transfer Protocol (HTTP)—also called "web-based email"
  - Messaging Application Programming Interface (MAPI)—used with Outlook in conjunction with a Microsoft Exchange Server mail server; very close to IMAP but has extended features to interact with other applications

More organizations are moving their email systems to the cloud. This essentially outsources many of the maintenance and security management issues associated with maintaining email servers and shifts expenditures from capital investments to operational expenditures. It also provides additional scalability and availability, which would be more difficult to achieve in smaller IT operations. However, the IS auditor must be mindful of the regulatory requirements of his or her organization. For example, for government agencies, if a foreign national uses a certain type of email that contains restricted data, this may be in violation of a law or regulation.

Security Issues of Email

According to J. Klensin and the Network Working Group, when reading the engineering standards established for email, these documents clearly indicate the following:
- SMTP mail is inherently insecure; a low level of complexity and skill is needed to perform a man-in-the-middle attack between receiving and relaying SMTP servers to then spoof legitimate email traffic.
- Real mail security lies only in end-to-end methods involving the message bodies, such as those that use digital signatures or integrity checks provided, at the transport level.

Some other security issues involved in emails are as follows:
- Phishing and spear phishing are electronic social engineering attacks that have become exceedingly sophisticated and can only be addressed through security awareness training.
- Flaws in the configuration of the mail server application may be used as the means of compromising the underlying server and the attached network.
- Denial-of-service (DoS) attacks may be directed to the mail server, denying or hindering valid users from using the mail server.
- Sensitive information transmitted unencrypted between mail server and email client may be intercepted.
- Information within the email may be altered at some point between the sender and recipient.
- Viruses and other types of malicious code may be distributed throughout an organization via email.
- Users may send inappropriate, proprietary or other sensitive information via email leading to a legal exposure.

Standards for Email Security
To improve email security, organizations should:
- Address the security aspects of the deployment of a mail server through maintenance and administration standards
- Ensure that the mail server application is deployed, configured and managed to meet the security policy and guidelines instituted by management
- Consider the implementation of encryption technologies to protect user authentication and mail data

As authorizers in many organizations use email to communicate approvals for business transactions (e.g., payroll run, journal entry posting, payment authorization), it is important to adopt a calibrated approach for ensuring the authenticity of emails based on an informed risk assessment. For example, organizations may require emails communicating authorizations to be digitally signed by the sender before they can be acted upon. In email security, a digital signature authenticates a transmission from a user in an untrusted network environment. A digital signature is a sequence of bits appended to a digital document. Like a handwritten signature, its authenticity can be verified. Unlike a handwritten signature, it is unique to the document being signed. Digital signatures are another application of public key cryptography. Digital signatures are a good method of securing email transmissions because:
- The signature cannot be forged.
- The signature is authentic and encrypted.
- The signature cannot be reused (a signature on one document cannot be transferred to another document).
- The signed document cannot be altered; any alteration to the document (whether or not it has been encrypted) renders the signature invalid.

There are two different types of encryption techniques used to ensure security. Messages can be secured with a single, bidirectional (encrypt/decrypt), secret, symmetric key system using Advanced Encryption Standard (AES) or with an asymmetric key system using pairs of unidirectional (only encrypt or decrypt), complementary keys, such as the public key management system RSA, a public key cryptosystem, was developed by R. Rivest, A. Shamir and L. Adleman and is often used (see section 5.4.5 Encryption).

If the email transmission is secured with use of a symmetric key on the receiver's end, the user needs to know the single secret key to decrypt the message. If the transmission is secured with an asymmetric key system using a public key, the user at the receiving end needs to use the private key to decrypt the message, as well as a digital signature verification program to verify the signature. Digital signatures are based on a procedure called message digesting, which computes a short, fixed-length number called a digest for any message of any length. Several different messages may have the same digest, but it is extremely difficult to produce any of them from the digest. A message digest is a cryptographically strong, one-way hash function of the message. It is similar to a checksum in that it compactly represents the message and is used to detect changes in the message. The message digest authenticates the user's message in such a way that if it were altered, the message would be considered corrupted.

Organizations should employ their network infrastructure to protect their mail server(s) through appropriate use of firewalls, routers and IDSs. (See chapter 5 Protection of Information Assets, for more information.)

3.7.6 POINT-OF-SALE SYSTEMS

Point-of-sale (POS) systems enable the capture of data at the time and place that sales transactions occur. The most common payment instruments to operate with POS are credit and debit cards, which are associated with bank accounts. POS terminals may have attached peripheral equipment—such as optical scanners to read bar codes and magnetic card readers for credit or debit cards or electronic readers for smart cards—to improve the efficiency and accuracy of the transaction recording process.

POS systems may be online to a central computer owned by a financial institution or a third-party administrator or may use local processors/microcomputers owned by a business to hold the transactions for a specified period, after which they are sent to the main computer for batch processing.

It is most important for the IS auditor to determine whether any cardholder data is stored on the local POS system such as primary account numbers (PANs), personal identification numbers (PINs), etc. Any such information, if stored on the POS system, should be encrypted using strong encryption methods. Certain data can never be stored on these devices such as card verification value (CVV) numbers.

Further information on standards for POS systems can be obtained at *www.pcisecuritystandards.org*, which addresses information on Payment Card Industry Data Security Standards (PCI DSS), and at *www.emvco.com*, which addresses standards for smart cards using embedded microprocessor chips.

3.7.7 ELECTRONIC BANKING

Banking organizations have been remotely delivering electronic services to consumers and businesses for years. Electronic funds transfer (EFT) (including small payments and corporate cash management systems), publicly accessible automated machines for currency withdrawal and retail account management are global fixtures.

Continuing technological innovation and competition among existing banking organizations and new market entrants has allowed for a much wider array of electronic banking products and services for retail and wholesale banking customers. However, the increased worldwide acceptance of the Internet as a delivery channel for banking products and services provides new business opportunities as well as new risk.

Major risk associated with banking activities includes strategic, reputational, operational (including security—sometimes called transactional—and legal risk), credit, price, foreign exchange, interest rate and liquidity. Electronic banking (e-banking) activities do not raise risk that was not already identified in traditional banking, but e-banking increases and modifies some of types of traditional risk. The core business and the IT environment are tightly coupled, thereby influencing the overall risk profile of e-banking.

In particular, from the perspective of the IS auditor, the main issues are strategic, operational and reputational risk because these are directly related to threats to reliable data flow and operational risk and are certainly heightened by the rapid introduction and underlying technological complexity of electronic banking.

Risk Management Challenges in E-banking
E-banking presents a number of risk management challenges:
- The speed of change relating to technological and service innovation in e-banking is unprecedented. Currently, banks are experiencing competitive pressure to roll out new business applications in very compressed time frames. This competition intensifies the management challenge to ensure that

adequate strategic assessment, risk analysis and security reviews are conducted prior to implementing new e-banking applications.
- Transactional e-banking web sites and associated retail and wholesale business applications are typically integrated as much as possible with legacy computer systems to allow more straight-through processing of electronic transactions. Such straight-through automated processing reduces opportunities for human error and fraud inherent in manual processes, but it also increases dependence on sound system design and architecture as well as system interoperability and operational scalability.
- E-banking increases banks' dependence on information technology, thereby increasing the technical complexity of many operational and security issues and furthering a trend toward more partnerships, alliances and outsourcing arrangements with third parties such as ISPs, telecommunication companies and other technology firms.
- The Internet is ubiquitous and global by nature. It is an open network accessible from anywhere in the world by unknown parties. Messages are routed through unknown locations and via fast-evolving wireless devices. Therefore, the Internet significantly magnifies the importance of security controls, customer authentication techniques, data protection, audit trail procedures and customer privacy standards.

Risk Management Controls for E-banking

Effective risk management controls for electronic banking include the following 15 controls divided among three categories:

- **Board and management oversight:**
  1. Effective management oversight of e-banking activities
  2. Establishment of a comprehensive security control process
  3. Comprehensive due diligence and management oversight process for outsourcing relationships and other third-party dependencies
- **Security controls:**
  4. Authentication of e-banking customers
  5. Nonrepudiation and accountability for e-banking transactions
  6. Appropriate measures to ensure SoD
  7. Proper authorization controls within e-banking systems, databases and applications
  8. Data integrity of e-banking transactions, records and information
  9. Establishment of clear audit trails for e-banking transactions
  10. Confidentiality of key bank information
- **Legal and reputational risk management:**
  11. Appropriate disclosures for e-banking services
  12. Privacy of customer information
  13. Capacity, business continuity and contingency planning to ensure availability of e-banking systems and services
  14. Incident response planning
  15. Compliance to banking sector directives (e.g., Basel Accords)

3.7.8 ELECTRONIC FINANCE

Electronic finance (e-finance) is an integral element of the financial services industry and enables providers to emerge within and across countries, including online banks, brokerages and companies that allow consumers to compare financial services such as mortgage loans and insurance policies. Nonfinancial entities are also entering the market, including telecommunication and utility companies that offer payment and other services.

Advantages of this approach to consumers are:
- Lower costs
- Increased breadth and quality
- Widening access to financial services
- Asynchrony (time-decoupled)
- Atopy (location-decoupled)

By using credit scoring and other data mining techniques, providers can create and tailor products over the Internet without much human input and at a very low cost. Providers can better stratify their customer base through analysis of Internet-collected data and allow consumers to build preference profiles online. This not only permits personalization of information and services, it also allows more personalized pricing of financial services and more effective identification of credit risk. At the same time, the Internet allows new financial service providers to compete more effectively for customers because it does not distinguish between traditional brick-and-mortar providers of financial services and those without physical presence. All these forces are delivering large benefits to consumers at the retail and commercial levels. These mechanisms should be used within privacy law statements (regarding confidentiality and authorization) to gather diverse user information and set up profiles.

3.7.9 PAYMENT SYSTEMS

There are two types of parties involved in all payment systems—the issuers and the users. An issuer is an entity that operates the payment service. An issuer holds the items that the payments represent (e.g., cash held in regular bank accounts). The users of the payment service perform two main functions—making payments and receiving payments—and, therefore, can be described as a payer or a payee, respectively.

Electronic Money Model

The objective of electronic money systems is to emulate physical cash. An issuer attempts to do this by creating digital certificates, which are then purchased (or withdrawn) by the users who redeem (deposit) them with the issuer at a later date. In the interim, certificates can be transferred among users to trade for goods or services.

For the certificates to take on some of the attributes of physical cash, certain techniques are used so that when a certificate is deposited, the issuer cannot determine the original withdrawer of the certificate. This provides the electronic certificates with unconditional untraceability.

Electronic money systems can be hard to implement in practice due to the overheads of solving the "double spending" problem (i.e., preventing a user from depositing the same money twice).

Some advantages of electronic money systems are:
- The payer does not need to be online (generally) at the time of the purchase (because electronic money can be stored on the payer's computer).
- The payer can have unconditional untraceability (albeit at the expense of lost interest on deposits).

Electronic Checks Model
Electronic check systems model real-world checks quite well and are thus relatively simple to understand and implement. A user writes an electronic check, which is a digitally signed instruction to pay. This is transferred (in the course of making a purchase) to another user, who then deposits the electronic check with the issuer. The issuer will verify the payer's signature on the payment and transfer the funds from the payer's account to the payee's account.

Some advantages of electronic check systems are:
- Easy to understand and implement
- The availability of electronic receipts, allowing users to resolve disputes without involving the issuer
- No need for payer to be online to create a payment

These systems are usually fully traceable, which is an advantage for certain law enforcement, tax collection and marketing purposes, but a disadvantage for those concerned about privacy.

Electronic Transfer Model
Electronic transfer systems are the simplest of the three payment models. The payer simply creates a payment transfer instruction, signs it digitally and sends it to the issuer. The issuer then verifies the signature on the request and performs the transfer. This type of system requires the payer to be online, but not the payee.

Some advantages of electronic transfer systems are:
- Easy to understand and implement
- The payee does not need to be online—a considerable advantage in some circumstances (e.g., paying employee wages)

3.7.10 INTEGRATED MANUFACTURING SYSTEMS

Integrated manufacturing systems (IMS) have a long history and, accordingly, there has been quite a diverse group of models and approaches.

Some of the integrated manufacturing systems include bill of materials (BOM), BOM processing (BOMP), manufacturing resources planning (MRP), computer-assisted design (CAD), computer-integrated (or computer-intensive) manufacturing (CIM) and manufacturing accounting and production (MAP).

Original IMSs were based on BOM and BOMP and usually supported by a hierarchical DBMS.

Evolution toward further integration with other business functions (e.g., recording of raw materials, work-in-process and finished goods transactions, inventory adjustments, purchases, supplier management, sales, accounts payable, accounts receivable, goods received, inspection, invoices, cost accounting, maintenance, etc.) led to MRP (initially standing for material requirements processing, now for manufacturing resources planning), which is a family of widely used standards and standard-based packages.

MRP is a typical module of most ERP packages such as SAP or Oracle Financials, and is usually integrated in modern customer relationship management (CRM) and supply chain management (SCM) systems.

CAD, computer-assisted engineering (CAE) and CAM—the latter including computerized numeric control (CNC)—have led to CIM. CIM is frequently used to run huge lights-out plants, with a significant portion of consumer goods being manufactured in these environments.

The importance for the IS auditor lies in the high number of systems and applications using these technologies. The larger the scale of integration, the more auditor attention is required.

Highly integrated CIM projects require the same attention from the auditor as the ERPs previously mentioned in this chapter. They are major undertakings that should be based on comprehensive feasibility studies and subject to top management approval and close supervision.

Continuity planning is also a primary area that should be reviewed by the IS auditor.

3.7.11 ELECTRONIC FUNDS TRANSFER

The method of payment plays a significant role in the relationship between seller and buyer. The underlying goal of the automated environment is to wring out costs inherent in the business processes. EFT is the exchange of money via telecommunications without currency actually changing hands. In other words, EFT is the electronic transfer of funds between a buyer, seller and his/her respective financial institution. EFT refers to any financial transaction that transfers a sum of money from one account to another electronically. EFT allows parties to move money from one account to another account, replacing traditional check writing and cash collection procedures. EFT services have been available for two decades. With the increased interest in Internet business, more and more consumers and businesses have begun to utilize EFT services. In the settlement between parties, EFT transactions usually function via an internal bank transfer from one party's account to another or via a clearinghouse network. Usually, transactions originate from a computer at one institution (location) and are transmitted to a computer at another institution (location) with the monetary amount recorded in the respective organization's accounts. Because of the potential high volume of money being exchanged, these systems may be in an extremely high-risk category. Therefore, access security and authorization of processing are important controls. Regarding EFT transactions, central bank requirements should be reviewed for application in these processes.

Controls in an EFT Environment
Because of the potential high volume of money being exchanged, these systems may be in an extremely high-risk category and security in an EFT environment becomes extremely critical.

Security includes the methods used by the customer to gain access to the system, the communications network, and the host or application processing site. Individual consumer access to the EFT system may be controlled by a plastic card and a PIN or by other means that bypass the need for a card. The IS auditor should review the physical security of unissued plastic cards, the procedures used to generate PINs, the procedures used to issue cards and PINs, and the conditions under which the consumer uses the access devices.

Security in an EFT environment ensures that:
- All the equipment and communication linkages are tested to effectively and reliably transmit and receive data

- Each party uses security procedures that are reasonably sufficient for affecting the authorized transmission of data and for protecting business records and data from improper access
- There are guidelines set for the receipt of data and to ensure that the receipt date and time for data transmitted are the date and time the data have been received
- On receipt of data, the receiving party will immediately transmit an acknowledgment or notification to communicate to the sender that a successful transmission occurred
- Data encryption standards are set
- Standards for unintelligible transmissions are set
- Regulatory requirements for enforceability of electronic data transmitted and received are explicitly stated

The IS auditor should ensure that reasonable authentication methods are required for access to EFT systems. The communications network should be designed to provide maximum security. Data encryption is recommended for all transactions; however, the IS auditor should determine any conditions under which the PIN might be accessible in a clear mode.

An EFT switch involved in the network is also an audit concern. An EFT switch is the facility that provides the communication linkage for all equipment in the network. The IS auditor should review the contract with the switch and the third-party audit of the switch operations. If a third-party audit has not been performed, the auditor should consider visiting the switch location.

At the application processing level, the IS auditor should review the interface between the EFT system and the application systems that process the accounts from which funds are transferred. Availability of funds or adequacy of credit limits should be verified before funds are transferred. Unfortunately, this is not always the case. Because of the penalties for failure to make a timely transfer, the IS auditor should review backup arrangements or other methods used to ensure continuity of operations. Because EFT reduces the flow of paper and consequently reduces normal audit trails, the IS auditor should determine that alternative audit trails are available.

3.7.12 AUTOMATED TELLER MACHINE

An ATM is a specialized form of the POS terminal that is designed for the unattended use by a customer of a financial institution. These machines customarily allow a range of banking and debit operations—especially financial deposits and cash withdrawals. ATMs are usually located in uncontrolled areas to facilitate easy access to customers after hours. This facility can be within a bank, across local banks and in banks outside a region. They are becoming known as retail EFT networks, transferring information and money over communication lines. Therefore the system must provide high levels of logical and physical security for both the customer and the machinery. The ATM architecture has a physical network layer, a switch and a communication layer connecting the various ATM POS terminals.

Recommended internal control guidelines for ATMs, apart from what has been provided for any EFT, include the following:
- Written policies and procedures covering personnel, security controls, operations, disaster recovery credit and check authorization, floor limits, override, settlement, and balancing
- Reconciliation of all general ledger accounts related to retail EFTs and review of exception items and suspense accounts
- Procedures for PIN issuance and protection during storage
- Procedures for the security of PINs during delivery and the restriction of access to a customer's account after a small number of unsuccessful attempts
- Systems should be designed, tested and controlled to preclude retrieval of stored PINs in any nonencrypted form. Application programs and other software containing formulas, algorithms and data used to calculate PINs must be subject to the highest level of access for security purposes.
- Controls over plastic card procurement should be adequate with a written agreement between the card manufacturer and the bank that details control procedures and methods of resolution to be followed if problems occur.
- Controls and audit trails of the transactions that have been made in the ATM. This should include internal registration in the ATM, either in internal paper or digital media, depending on regulation or laws in each country and on the hosts that are involved in the transaction.

Audit of ATMs

To perform an audit of ATMs, the IS auditor should:
- Review physical security to prevent introduction of malware (such as Tyupkin)
- Review measures to establish proper customer identification and maintenance of their confidentiality
- Review file maintenance and retention system to trace transactions
- Review exception reports to provide an audit trail
- Review daily reconciliation of ATM transactions including:
  - Review SoD in the opening of ATM and recount of deposit
  - Review the procedures made for the retained cards
- Review encryption key change management procedures
  - Physical security measures to ensure security of the ATM and the money contained in the ATM
  - Review of ATM card slot, key pad and enclosure to prevent skimming of card data and capture of PIN during entry

3.7.13 INTERACTIVE VOICE RESPONSE

In telephony, interactive voice response (IVR), is a phone technology that allows a computer to detect voice and touch tones using a normal phone call. The caller uses the telephone keypad to select from preset menu choices provided by the IVR. The IVR system then responds with pre-recorded or dynamically generated audio to further direct callers or route the caller to a customer service representative. IVR systems can be used to control almost any function where the interface can be broken down into a series of simple menu choices. IVR systems generally scale well to handle large call volumes. Controls over such systems must be in place to prevent unauthorized individuals from entering system—level commands that may permit them to change or rerecord menu options.

3.7.14 PURCHASE ACCOUNTING SYSTEM

Financial transactions frequently go through more than one system when processed. In a department store, a sale is first processed in the sales accounting system, then processed by the accounts receivable system (if the purchase was by credit card) and, for either cash or credit sales, through the inventory system (when they are linked). That same sale might trigger the purchase accounting system to replace depleted inventory. Eventually the transactions become part of the general ledger system because all transactions are recorded somewhere in that system. For the integration of systems to be effective,

processing of transactions must be complete, accurate and timely. If it is not, a ripple effect impairs the integrity of the data.

Purchase accounting systems process the data for purchases and payments. Because purchases automatically lead to payments, if purchases are properly contracted, partial control over payments exists. Additional controls over payments are still needed to ensure that each payment was made for goods and services received, that the same purchases were not paid for twice and that they were, indeed, paid. Most purchase accounting systems perform three basic accounting functions:

1. **Accounts payable processing**—Recording transactions in the accounts payable records
2. **Goods received processing**—Recording details of goods received but not yet invoiced
3. **Order processing**—Recording goods ordered but not yet received

The computer may be involved in each of these activities, and the extent to which they are computerized determines the complexity of the purchase accounting system.

3.7.15 IMAGE PROCESSING

Some of the many algorithms used in image processing include convolution (on which many others are based), fast Fourier transform (FFT), discrete cosine transform (DCT), thinning (or skeletonization), edge detection and contrast enhancement. These are usually implemented in software but may also use special-purpose hardware for speed.

An imaging system stores, retrieves and processes graphic data, such as pictures, charts and graphs, instead of or in addition to text data. The storage capacities must be enormous and most image systems include optical disk storage. In addition to optical disks, the systems include high-speed scanning, high-resolution displays, rapid and powerful compression, communications functions, and laser printing. The systems include techniques that can identify levels of shades and colors that cannot be differentiated by the human eye. These systems are expensive and companies do not invest in them lightly.

Most businesses that perform image processing obtain benefits from using the imaging system. Examples of potential benefits are:

- Item processing (e.g., signature storage and retrieval)
- Immediate retrieval via a secure optical storage medium
- Increased productivity
- Improved control over paper files
- Reduced deterioration due to handling
- Enhanced disaster recovery procedures

Imaging systems are the fastest growth area of the micrographics industry and are an outgrowth of microfilm and microfiche, which have, in the past, been heavily used in paper-intensive fields such as insurance and banking. Not surprisingly, these same fields were the first to incorporate imaging systems into their standard operations. The replacement of paper documents with electronic images can have a significant impact on the way that an institution does business. Many of the traditional audit and security controls for paper-based systems may be reduced or absent in electronic documentation workflow. New controls must be developed and designed into the automated process to ensure that information image files cannot be altered, erased or lost.

Risk areas that management should address when installing imaging systems and that IS auditors should be aware of when reviewing an institution's controls over imaging systems include:

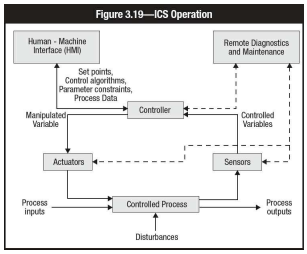
- **Planning**—The lack of careful planning in selecting and converting paper systems to document imaging systems can result in excessive installation costs, the destruction of original documents and the failure to achieve expected benefits. Critical issues include converting existing paper storage files and integration of the imaging system into the organization workflow and electronic media storage to meet audit and document retention legal requirements.
- **Audit**—Imaging systems may change or eliminate the traditional controls as well as the checks and balances inherent in paper-based systems. Audit procedures may have to be redesigned and new controls designed into the automated process.
- **Redesign of workflow**—Institutions generally redesign or reengineer workflow processes to benefit from imaging technology.
- **Scanning devices**—Scanning devices are the entry point for image documents and a significant risk area in imaging systems. Scanning operations can disrupt workflow if the scanning equipment is not adequate to handle the volume of documents or the equipment breaks down. The absence of controls over the scanning process can result in poor quality images, improper indexing, and incomplete or forged documents being entered into the system. Factors that should be considered in an imaging system are quality control over the scanning and indexing process, the scanning rate of the equipment, the storage of images, equipment backup, and the experience level of personnel scanning the document. Procedures should be in place to ensure that original documents are not destroyed before determining that a good image has been captured.
- **Software security**—Security controls over image system documents are critical to protect institutions and customer information from unauthorized access and modifications. The integrity and reliability of the imaging system database is related directly to the quality of controls over access to the system.
- **Training**—Inadequate training of personnel scanning the documents can result in poor-quality document images and indexes, and the early destruction of original documents. The installation and use of imaging systems can be a major change for department personnel. They must be trained adequately to ensure quality control over the scanning and storage of imaging documents as well as the use of the system to maximize the benefits of converting to imaging systems.

3.7.16 INDUSTRIAL CONTROL SYSTEMS

Industrial control system (ICS) is a general term that encompasses several types of control systems, including supervisory control and data acquisition (SCADA) systems, distributed control systems (DCS), and other control system configurations such as programmable logic controllers (PLC), often found in the industrial sectors and critical infrastructures.

Figure 3.19 provides a high-level overview of typical ICS process flows.

Figure 3.19—ICS Operation



Source: NIST; *NIST SP 800-82: Guide to Industrial Control Systems (ICS) Security*, USA, 2011

Risk Factors

Based on the criticality that ICSs have on manufacturing, chemical processes and, more important, critical infrastructure (energy generation, transmission and control, water treatment, etc.), there are key risk factors that the IS auditor must consider:

- Blocked or delayed flow of information through ICS networks, which could disrupt ICS operation
- Unauthorized changes to instructions, commands or alarm thresholds, which could damage, disable or shut down equipment, create environmental impacts and/or endanger human life
- Inaccurate information sent to system operators, either to disguise unauthorized changes or to cause the operators to initiate inappropriate actions, which could have various negative effects
- ICS software or configuration settings modified or ICS software infected with malware, which could have various negative effects
- Interference with the operation of safety systems, which could endanger human life

Typical Controls

To address risk, preventive, detective and corrective controls need to be considered within the administrative, operational and technical implementation of ICS:

Restricting logical access to the ICS network and network activity. This includes using a demilitarized zone (DMZ) network architecture with firewalls to prevent network traffic from passing directly between the corporate and ICS networks and having separate authentication mechanisms and credentials for users of the corporate and ICS networks. The ICS should also use a network topology that has multiple layers, with the most critical communications occurring in the most secure and reliable layer.

- Restricting physical access to the ICS network and devices. Unauthorized physical access to components could cause serious disruption of the ICS's functionality. A combination of physical access controls should be used, such as locks, card readers and/or guards.
- Protecting individual ICS components from exploitation. This includes deploying security patches in as expeditious a manner as possible, after testing them under field conditions; disabling all unused ports and services; restricting ICS user privileges to only those that are required for each person's role; tracking and monitoring audit trails; and using security controls such as antivirus software and file integrity checking software, where technically feasible, to prevent, deter, detect and mitigate malware.
- Maintaining functionality during adverse conditions. This involves designing the ICS so that each critical component has a redundant counterpart. Additionally, if a component fails, it should fail in a manner that does not generate unnecessary traffic on the ICS or other networks, or does not cause another problem elsewhere, such as a cascading event.
- Restoring system after an incident. Incidents are inevitable and an incident response plan is essential. A major characteristic of a good security program is how quickly a system can be recovered after an incident has occurred.

3.7.17 ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

Artificial intelligence (AI) is the study and application of the principles by which:

- Knowledge is acquired and used.
- Goals are generated and achieved.
- Information is communicated.
- Collaboration is achieved.
- Concepts are formed.
- Languages are developed.

AI fields include, among others:

- Expert systems
- Natural and artificial (such as programming) languages
- Neural networks
- Intelligent text management
- Theorem proving
- Abstract reasoning
- Pattern recognition
- Voice recognition
- Problem solving
- Machine translation of foreign languages

Two main programming languages that have been developed for AI are Lisp and Prolog.

Expert systems are an area of AI and perform a specific function or are prevalent in certain industries. An expert system allows the user to specify certain basic assumptions or formulas and then uses these assumptions or formulas to analyze arbitrary events. Based on the information used as input to the system, a conclusion is produced.

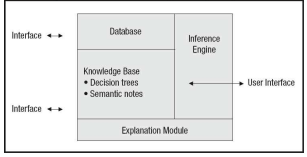
The use of expert systems has many potential benefits within an organization including:

- Capturing the knowledge and experience of individuals
- Sharing knowledge and experience

- Enhancing personnel productivity and performance
- Automating highly (statistically) repetitive tasks (help desk, score credits, etc.)
- Operating in environments where a human expert is not available (e.g., medical assistance on board of a ship, satellites, etc.)

Expert systems are comprised of the primary components shown in **figure 3.20**, called shells, when they are not populated with particular data, and the shells are designed to host new expert systems.

Figure 3.20—Artificial Intelligence Expert Systems



Key to the system is the knowledge base (KB), which contains specific information or fact patterns associated with particular subject matter and the rules for interpreting these facts. The KB interfaces with a database in obtaining data to analyze a particular problem in deriving an expert conclusion. The information in the KB can be expressed in several ways:
- **Decision trees**—Using questionnaires to lead the user through a series of choices, until a conclusion is reached. Flexibility is compromised because the user must answer the questions in an exact sequence.
- **Rules**—Expressing declarative knowledge through the use of if-then relationships. For example, if a patient's body temperature is over 39°C (102.2°F) and his/her pulse is under 60, then the patient might be suffering from a certain disease.
- **Semantic nets**—Consist of a graph in which the nodes represent physical or conceptual objects and the arcs describe the relationship between the nodes. Semantic nets resemble a data flow diagram and make use of an inheritance mechanism to prevent duplication of data.

Additionally, the inference engine shown is a program that uses the KB and determines the most appropriate outcome based on the information supplied by the user. In addition, an expert system includes the following components:
- **Knowledge interface**—Allows the expert to enter knowledge into the system without the traditional mediation of a software engineer
- **Data interface**—Enables the expert system to collect data from nonhuman sources, such as measurement instruments in a power plant

An explanation module that is user-oriented in addressing the problem is analyzed, and the expert conclusion reached is also provided.

Expert systems are gaining acceptance and popularity as audit tools. Specifically related to IS auditing, expert systems have been developed to facilitate the IS auditor in auditing such areas as OSs, online software environments, access control products and microcomputer environments. These tools can take the form of a series of well-designed questionnaires or actual software that integrates and reports on system parameters and data sets. Other accounting- and auditing-related applications for expert systems include audit planning, internal control analysis, account attribute analysis, quality review, accounting decisions, tax planning and user training.

Consistent with standard systems development methodologies, stringent change control procedures should be followed because the basic assumptions and formulas may need to be changed as more expertise is gained. As with other systems, access should be on a need-to-know basis.

The IS auditor should be knowledgeable about the various AI and expert system applications used within the organization.

The IS auditor needs to be concerned with the controls relevant to these systems when used as an integral part of an organization's business process or mission critical functions, and the level of experience or intelligence used as a basis for developing the software. This is critical because errors produced by AI systems may have a more severe impact than those produced by traditional systems. This is true especially of intelligent systems that facilitate health care professionals in the diagnosis and treatment of injuries and illnesses. Error loops/routines should be designed into these systems.

Specifically, the IS auditor should:
- Understand the purpose and functionality of the system.
- Assess the system's significance to the organization and related businesses processes as well as the associated potential risk.
- Review the adherence of the system to corporate policies and procedures.
- Review the decision logic built into the system to ensure that the expert knowledge or intelligence in the system is sound and accurate. The IS auditor should ensure that the proper level of expertise was used in developing the basic assumptions and formulas.
- Review procedures for updating information in the KB.
- Review security access over the system, specifically the KB.
- Review procedures to ensure that qualified resources are available for maintenance and upgrading.

3.7.18 BUSINESS INTELLIGENCE

Business intelligence (BI) is a broad field of IT that encompasses the collection and analysis of information to assist decision making and assess organizational performance.

Investments in BI technology can be applied to enhance understanding of a wide range of business questions. Some typical areas in which BI is applied for measurement and analysis purposes include:
- Process cost, efficiency and quality
- Customer satisfaction with product and service offerings
- Customer profitability, including determination of which attributes are useful predictors of customer profitability
- Staff and business unit achievement of key performance indicators
- Risk management (e.g., by identifying unusual transaction patterns and accumulation of incident and loss statistics)

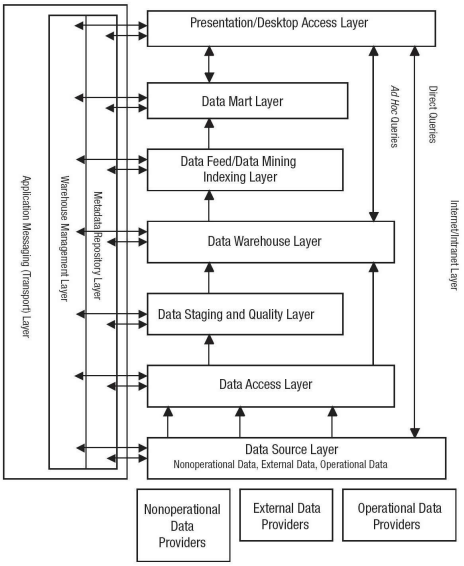
The interest in BI as a distinct field of IT activity is being spurred by a number of factors:
- **The increasing size and complexity of modern organizations**—The result is that even fundamental business questions cannot be properly answered without establishing serious BI capability.

- **Pursuit of competitive advantage**—Most organizations have, for many years, automated their basic, high-volume activities. Significant organizationwide IT investment such as ERP systems is now common place. Many companies have or are now investing in Internet technology as a means of distributing product/service and supply chain integration. However, utilization of IT to maintain and extend a firm's knowledge capital represents a new opportunity to use technology to gain an advantage over competitors.
- **Legal requirements**—Legislation such as the US Sarbanes-Oxley Act and the US Patriot Act exist to enforce the need for companies to have an understanding of the "whole of business." Financial institutions must now be able to report on all accounts/instruments that their customers have and all transactions against those accounts/instruments, including any suspicious transaction patterns.

To deliver effective BI, organizations need to design and implement (progressively, in most cases) a data architecture. A complete data architecture consists of two components:
- The enterprise data flow architecture (EDFA)
- A logical data architecture

An example of optimized enterprise data flow architecture is depicted in **figure 3.21**. Explanations of the various layers/components of this data flow architecture follow:
- **Presentation/desktop access layer**—This is where end users directly deal with information. This layer includes familiar desktop tools such as spreadsheets, direct querying tools, reporting and analysis suites offered by vendors such as Cognos and Business Objects, and purpose-built applications such as balanced scorecards (BSCs) and digital dashboards. Power users will have the ability to build their own queries and reports while other users will interact with the data in predefined ways. Increasingly, users are being provided with more than static reporting capabilities by enhancing reports with parameterization and drill-down capabilities and presenting data in visual formats as a supplement or replacement of textual/tabular data presentation.

Figure 3.21—Data Flow Architecture Sample



- **Data source layer**—Enterprise information derives from a number of sources:
  - Operational data—Data captured and maintained by an organization's existing systems and usually held in system-specific databases or possibly flat files
  - External data—Data provided to an organization by external sources. This could include data such as customer demographics and market share information.
  - Nonoperational data—Information needed by end users that is not currently maintained in a computer-accessible format
- **Core data warehouse**—A core data warehouse (DW) is where all (or at least the majority of) the data of interest to an organization is captured and organized to assist reporting and analysis. DWs are normally instituted as large relational databases. While there is not unanimous agreement, many pundits suggest the warehouse should hold fully normalized data to give it the flexibility to deal with complex and changing business structures. A properly constituted DW should support three basic forms of inquiry:
  - Drilling up and drilling down—Using dimensions of interest to the business, it should be possible to aggregate data (e.g., sum store sales to get region sales and ultimately national sales) as well as drill down (e.g., break store sales down to counter sales). Attributes available at the more granular levels of the warehouse can also be used to refine the analysis (e.g., analyze sales by product).
  - Drill across—Use common attributes to access a cross-section of information in the warehouse such as sum sales across all product lines by customer and groups of customers according to length of association with the company (and/or other attribute of interest)
  - Historical analysis—The warehouse should support this by holding historical, time-variant data. An example of historical analysis would be to report monthly store sales and then repeat the analysis using only customers who were preexisting at the start of the year in order to separate the effect of new customers from the ability to generate repeat business with existing customers.
- **Data mart layer**—Data marts represent subsets of information from the core DW selected and organized to meet the needs of a particular business unit or business line. Data marts may be relational databases or some form of online analytical processing (OLAP) data structure (also known as a data cube). OLAP technologies and some variants (e.g., relational OLAP [ROLAP]), allow users to "slice and dice" data presented in terms of standardized measures (i.e., numerical facts) and dimensions (i.e., business hierarchies). Data marts have a simplified structure compared to the normalized DW.
- **Data staging and quality layer**—This layer is responsible for data copying, transformation into DW format and quality control. It is particularly important that only reliable data get loaded to the core DW. This layer needs to be able to deal with problems periodically thrown up by operational systems such as changes to account number formats and reuse of old account and customer numbers (when the DW still holds information on the original

entity).

- **Data access layer**—This layer operates to connect the data storage and quality layer with data stores in the data source layer and, in the process, avoiding the need to know exactly how these data stores are organized. Technology now permits structured query language (SQL) access to data even if it is not stored in a relational database.
- **Data preparation layer**—This layer is concerned with the assembly and preparation of data for loading into data marts. The usual practice is to precalculate the values that are loaded into OLAP data repositories to increase access speed. Specialist data mining also normally requires preparation of data. Data mining is concerned with exploring large volumes of data to determine patterns and trends of information. Data mining often identifies patterns that are counterintuitive due to the number and complexity of data relationships. Data quality needs to be very high to not corrupt the results.
- **Metadata repository layer**—Metadata are data about data. The information held in the metadata layer needs to extend beyond data structure names and formats to provide detail on business purpose and context. The metadata layer should be comprehensive in scope, covering data as they flow between the various layers, including documenting transformation and validation rules. Ideally, information in the metadata layer can be directly sourced by software operating in the other layers, as required.
- **Warehouse management layer**—The function of this layer is the scheduling of the tasks necessary to build and maintain the DW and populate data marts. This layer is also involved in the administration of security.
- **Application messaging layer**—This layer is concerned with transporting information between the various layers. In addition to business data, this layer encompasses generation, storage and targeted communication of control messages.
- **Internet/intranet layer**—This layer is concerned with basic data communication. Included here are browser-based user interfaces and TCP/IP networking.

The construction of the logical data architecture for an enterprise is a major undertaking that would normally be undertaken in stages. One reason for separating logical data model determination by business domain is that different parts of large business organizations will often deal with different transaction sets, customers and products.

Ultimately, the data architecture needs to be structured to accommodate the needs of the organization in the most efficient manner. Factors to consider include the types of transactions in which the organization engages, the entities that participate in or form part of these transactions (e.g., customers, products, staff and communication channels), and the dimensions (hierarchies) that are important to the business (e.g., product and organization hierarchies).

With modern DWs, storage capacity is not really an issue. Therefore, the goal should be to obtain the most granular or atomic data possible. The lowest level data are most likely to have attributes that can be used for analysis purposes that would be lost if summarized data are loaded.

Various analysis models used by data architects/analysts follow:

- **Context diagrams**—Outline the major processes of an organization and the external parties with which the business interacts.
- **Activity or swim-lane diagrams**—Deconstruct business processes.
- **Entity relationship diagrams**—Depict data entities and how they relate. These data analysis methods obviously play an important part in developing an enterprise data model. However, it is also crucial that knowledgeable business operatives are involved in the process. This way proper understanding can be obtained of the business purpose and context of the data. This also mitigates the risk of the replication of suboptimal data configurations from existing systems and databases into the DW.

Business Intelligence Governance

To maximize the value an organization obtains from its BI initiatives, an effective BI governance process needs to be in place.

An important part of the governance process involves determining which BI initiatives to fund, what priority to assign to initiatives and how to measure their ROI. This is particularly important because the investment needed to build BI infrastructure, such as a DW, is considerable. Additionally, the scope and complexity of an organizationwide DW means that, realistically, it must be built in stages.

A recommended practice in the area of BI funding governance is to establish a business/IT advisory team that allows different functional perspectives to be represented, recommends investment priorities and establishes cross-organizational benefit measures. Final funding decisions should rest with a technology steering committee that comprises senior management.

A further important part of overall BI governance is data governance. Aspects to be considered here include establishing standard definitions for data, business rules and metrics, identifying approved data sources, and establishing standards for data reconciliation and balancing.

3.7.19 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 are:

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

Efficiency vs. Effectiveness

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.

Decision Focus

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 DSS are nothing more than a DSS generator (tools with which to construct a DSS).

DSS Frameworks

Frameworks are generalizations about a field that help put many specific cases and ideas into perspective. The G. Gorry-M.S. Morton framework is the most complete knowledge- and system-control-related IS model, and it is based on problem classification into structured and unstructured types as well as the time horizon of the decisions. This framework characterizes DSS activities along two dimensions:

1. The degree of structure in the decision process being supported
2. The management level at which decision making takes place

This framework also portrays all IS efforts as addressing distinct types of problems, depending on the above two factors.

The management-level dimension is broken into three parts:

1. Operational control
2. Management control
3. Strategic planning

The decision-structure dimension is also broken into three parts:

1. Structured
2. Semistructured
3. Unstructured

The degree to which a problem or decision is structured corresponds roughly to the extent to which it can be automated or programmed.

Another DSS framework is the Sprague-Carlson framework that is initiated with an effort to create family trees—which is a generalization of the structure of a DSS. This framework suggests that every DSS has data, a model and a dialog generator subsystem. This framework emphasizes the importance of data management in DSS work. This framework also stresses the importance of interactive user interfaces in a DSS. The generation and management of these interfaces require appropriate software and hardware—the dialog management system. In general, a system must offer more than one interface and might need to provide a tailored interface for each user.

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 something 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

Some common trends in DSS usage include:
- The need for more accurate information by managers is an important motivator for DSS development.
- Few DSS evaluations use the traditional cost-benefit analysis.
- End users are usually the motivators in developing a DSS.
- Development staff members are drawn largely from functional area staff or the planning department, not from the IT department.
- Users perceive flexibility as the most important factor influencing the system's success.
- Few DSS projects are being developed today for third-generationsoftware facilities. User-oriented, 4GLs and planning languages predominate.
- Planning, evaluation and training for DSS projects traditionally have been performed quite poorly.

DSS Common Characteristics
Some of the common characteristics of DSSs 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; i.e., determining what is the end result of a changing variable or variables)
- Sensitivity analysis
- Goal-seeking
- Excellent graphic presentations
- Dynamic graphic, data editing
- Simulation

DSS Trends
DSS trends include:
- Gradual improvement and sharpening of skills in the development and implementation of a traditional DSS
- Advances in database and graphics capabilities for microcomputers
- Exploratory work in such fields as expert systems, a DSS to support group decision making and visual interactive modeling

3.7.20 CUSTOMER RELATIONSHIP MANAGEMENT

The customer-driven business trend is to be focused on the wants and needs of the customers. With the customer's expectations constantly increasing, these objectives are becoming more difficult to achieve. This emphasizes the importance of focusing on information relating to transaction data, preferences, purchase patterns, status, contact history, demographic information and service trends of customers rather than on products.

All these factors lead CRM, which is an optimum combination of strategy, tactics, processes, skill sets and technology. CRM has become a strategic success factor for all types of business, and its proficiency has a significant impact on profitability.

The customer expectations are increasing tremendously and that, in turn, raises the expectation of service levels. Therefore, the customer-centered applications focus on CRM processes emphasizing the customer, rather than marketing, sales or any other function. The new business model will have an

integration of telephony, web and database technologies, and interenterprise integration capabilities. Also, this model spreads to the other business partners who can share information, communicate and collaborate with the organization with the seamless integration of web-enabled applications and without changing their local network and other configurations.

It is possible to distinguish between operational and analytical CRM. Operational CRM is concerned with maximizing the utility of the customer's service experience while also capturing useful data about the customer interaction. Analytical CRM seeks to analyze information captured by the organization about its customers and their interactions with the organization into information that allows greater value to be obtained from the customer base. Among uses of analytical CRM are increasing customer product holdings or "share of customer wallet," moving customers into higher margin products, moving customers to lower-cost service channels, increasing marketing success rates, and making pricing decisions.

3.7.21 SUPPLY CHAIN MANAGEMENT

Supply chain management (SCM) is linking the business processes between the related entities such as the buyer and the seller. The link is provided to all the connected areas such as managing logistics and the exchange of information, services and goods among supplier, consumer, warehouse, wholesale/retail distributors and the manufacturer of goods.

SCM has become a focal point and is seen as a new area in strategic management because of the shift in the business scenario at the advent of global competition, proliferation of the Internet, the instantaneous transmission of information and web presence in all spheres of business activities. SCM is all about managing the flow of goods, services and information among suppliers, manufacturers, wholesalers, distributors, stores, consumers and end users.

SCM shifts the focus; all the entities in the supply chain can work collaboratively and in a real-time mode, thus reducing, to a great extent, the required available inventory. The just-in-time (JIT) concept becomes possible, and the cycle time is reduced with an objective toward reducing the unneeded inventory. Seasonal (e.g., both availability and demand) and regional (e.g., preferences in size, shape, quantity, etc.) factors are addressed.

Stock levels of nonmoving items are significantly reduced, and there is an automated flow of supply and demand. Also, the intrinsic costs and errors associated with the manual means such as fax, input of data, delay and inaccurate orders, can be avoided.

3.8 DEVELOPMENT METHODS

In the face of increasing system complexity and the need to implement new systems more quickly to achieve benefits before the business changes, software development practitioners have adopted new ways of organizing software projects that vary, or in some cases radically depart from, the traditional waterfall model previously described. In addition, there has been continued evolution in the thinking about how best to analyze, design and construct software systems and in the information technologies available to perform these activities.

This section describes different techniques of understanding, designing and constructing a software system. The choice of a particular method will be driven by considerations such as organizational policy, developer knowledge and preference, and the technology being used.

Please note that the selection of one of the methods described in this section is generally independent of the selection of a project organization model. An object-oriented approach to design and coding could be utilized on a project organized into distinct phases as in the waterfall model of software development just as it could be with an agile project where each short iteration delivers working software.

3.8.1 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 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 minispecifications.
- 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.

Auditors 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.

3.8.2 AGILE DEVELOPMENT

The term "agile development" refers to a family of similar development processes that espouse a nontraditional way of developing complex systems. One of the first agile processes, Scrum (a rugby analogy), emerged in the early 1990s.

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, achieving their objectives. Scrum is a project management approach that fits well with other agile techniques. Other agile processes have since emerged such as Extreme Programming (XP), Crystal, Adaptive Software Development, Feature Driven Development and Dynamic Systems Development Method. 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, as shown in **figure 3.22**. 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 of development in detail, rather than planning subsequent development phases far out in time.
- Agile development's adaptive approach to requirements does not emphasize managing a requirements baseline.
- Agile development's focus is to quickly prove an architecture by building actual functionality versus formally defining, early on, software and data architecture in increasingly more detailed models and descriptions.
- Agile development 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.
- Agile development does not emphasize defined and repeatable processes but instead performs and adapts its development based on frequent inspections.

3.8.3 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.

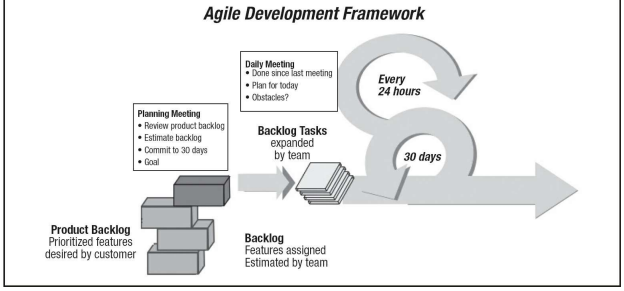
The initial emphasis during the development of the prototype is usually placed on the reports and screens, which are the system aspects most used by the end users. This allows the end user to see a working model of the proposed system within a short time.

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 not often 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.

Figure 3.22—Data Flow Architecture Sample



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 that come out of the traditional system development approach such as backup recovery, security and audit trails.

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 the IS auditor should be aware of the risk associated with prototyping, the IS auditor should also be aware that this method of system

development can provide the organization with significant time and cost savings.

3.8.4 RAPID APPLICATION DEVELOPMENT

RAD is a methodology that enables organizations 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 business process reengineering (BPR) attempts to convert an existing business process rather than make dynamic changes.

3.8.5 OBJECT-ORIENTED SYSTEM DEVELOPMENT

OOSD is the process of solution specification and modeling where 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 which considered 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. One can do OOSD while following any of the widely diverse set of software methodologies: waterfall, iterative, software engineering, agile and even pure hacking (and prototyping). 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.

The first objected-oriented language, Simula67, was released in 1967. Smalltalk emerged during the 1970s as the first commercial object-oriented language. Then followed a series of languages that were either object-oriented from inception (e.g., Eiffel) or had been modified to include object-oriented capabilities (e.g., C++, Object Pascal, and Ada95). The emergence of Java during the late 1990s provided a significant boost to the acceptance of object technology.

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 has been the decision by some of the major players in object-oriented development to join forces and merge their individual approaches into a unified approach using the Unified Modeling Language (UML). UML is a general-purpose notational language for specifying and visualizing complex software for large object-oriented projects. 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
- E-business applications
- Computer-aided software engineering (CASE) for software development
- Office automation for email and work orders
- Artificial intelligence
- CAM for production and process control

3.8.6 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 which programming language they were written in or what OS they are running). The basic types of components are:

- **In-process client components**—These components must run from within a container of some kind such as a web browser; they cannot run on their own.
- **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. These are initiated by remote procedure calls or some other kind of network call. Technologies supporting this include Microsoft's Distributed Component Object Model (DCOM), Object Management Group's Common Object Request Broker Architecture (CORBA) and Sun's Java through Remote Method Invocation (RMI) are sometimes referred to as distributed object technologies
- **In-process server components**—These components run on servers within containers. Examples include Microsoft's Transaction Server (MTS) and Oracle's JavaBeans.

Note: The CISA candidate will not be tested on vendor-specific products or services in the CISA exam.

A number of different component models have emerged. Microsoft has its Component Object Model (COM). MTS when combined with COM allows developers to create components that can be distributed in the Windows environment. COM is the basis for ActiveX technologies, with ActiveX Controls being among the most widely used components. Alternative component models include the CORBA Component Model and JavaBeans.

Please note that COM/DCOM, CORBA (itself a standard, not a specific product) and RMI are sometimes referred to as distributed object technologies. As the name suggests, they allow objects on distributed platforms to interact. These technologies, among others, are also termed middleware. Middleware is a broad term, but a basic definition is software that provides run-time services whereby programs/objects/components can interact with one another.

Tool developers are supporting one or another of these standards with powerful, visual tools now available for designing and testing component-based applications. Industry "heavyweights" such as Microsoft and IBM are supporting component-based development. Additionally, a growing number of commercially available application servers now support MTS or 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.

3.8.7 WEB-BASED APPLICATION DEVELOPMENT

Web-based application development is an important emerging 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 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 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 to avoid 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 that a common update process updates the multiple places required. Web services are intended to make this relatively easy to achieve.

Web application development is different than traditional third- or fourth-generation program developments in many ways—from the languages and programming techniques used, to the methodologies (or lack thereof) 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 had been a risk since computer programming was invented (for example, truncation issues with first generation computer programs), but 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. One's previous experience can be used to identify risk related to inadequate specifications, poor coding techniques, inadequate documentation, inadequate 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.

Standards for SOAP, WSDL and UDDI have been accepted by the World Wide Web Consortium. A number of current software products and development environments, including Microsoft's. Net family of products, support web services. However, some important standards, such as those addressing security and transaction management, are yet to be defined. Other issues, such as charging for use of commercially developed web services, also need to be addressed.

3.8.8 SOFTWARE REENGINEERING

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 now available to support this process. Typical methodologies used in software reengineering generally fall into the following categories:

- Business process reengineering (BPR) is the thorough analysis and significant redesign of business processes and management systems to establish a better performing structure, 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 (see section 3.7.4 Controls in the EDI Environment) leveraging RACI (responsible, accountable, consulted and informed) charts and UML modeling.

3.8.9 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 several 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

The IS auditor should be aware of the following risk:

- Software license agreements often contain clauses prohibiting the licensee from reverse engineering the software so that any trade secrets or programming techniques are not 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.

3.9 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 is a critical task for an IT department. Its 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, 24/7 availability and many others 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.

In section 3.9.1 Project Phases of Physical Architecture Analysis, steps are listed to arrive at the choice of a good physical architecture and the way to define a possible road map for supporting the migration of the technical architecture to a new one to reach the following goals:

- To successfully analyze the existing architecture (including data flow analysis defining all data being received, processed, stored and transmitted)
- To design a new architecture that takes into account the existing architecture and a company's particular constraints/requirements, such as:
  - Scalability and Interoperability to handle all data currently and having the potential of being received, processed, stored and transmitted
  - Reduced costs
  - Increased functionality
  - Minimum impact on daily work

  - Security and confidentiality issues
  - Progressive migration to the new architecture
- To write the functional requirements of this new architecture
- To develop a proof of concept (POC) based on these functional requirements:
  - To characterize price, functionality and performance
  - To identify additional requirements that will be used later

The resulting requirements will be documented in specifications and drawings describing the reference infrastructure that will be used by all projects downstream. With these requirements in hand, these projects can begin to start implementation.

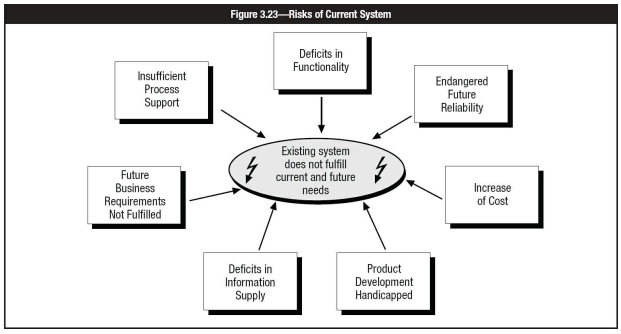
The requirements are validated using a proof of concept. The proof of concept is a test-bed implementation of the physical architecture. It saves money because any problems are detected and corrected early, when they are cheaper to correct, and it gives confidence to the teams that the requirements correctly instruct potential vendors on the requirements they have to meet.

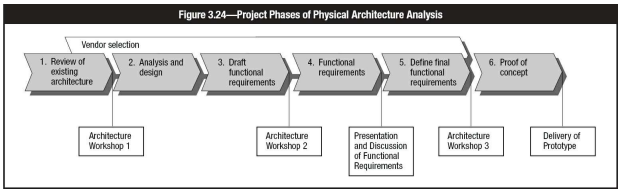
The main objective of section 3.9.2 Planning Implementation of Infrastructure, is to plan the physical implementation of the required technical infrastructure to set up the future environment (normally production, test and development environment). This task will cover the procurement activities such as contracting partners, setting up the SLAs, and developing installation plans and installation test plans. A well-designed selection process must be ensured, taking analytical results and intuition into account and guaranteeing alignment and commitment for implementation success. Due to the possible heterogeneous nature of the infrastructure found, it is necessary to develop a clear implementation plan (including deliverables, delivery times, test plans, etc.). It is also necessary to plan the coexistence of the old and new system, to avert possible mistakes during the installation and go-live phase (**figure 3.23**).

Thus, information and communication technologies (ICT) departments often face these requirements. The suggested solution must:
- 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

3.9.1 PROJECT PHASES OF PHYSICAL ARCHITECTURE ANALYSIS

Figure 3.24 shows the project phases to physical architecture analysis and, in the background, the time at which the vendor selection process may take place.





Review of Existing Architecture
To start the process, the latest documents about 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. This process is described in detail later in this chapter.

Writing Functional Requirements
After finishing the draft functional requirements and feeding the second part of this project, the functional requirements document is written, which 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 POC starts, although the planning of the POC starts after the second workshop. With the finished functional requirements, the proof of concept phase begins.

Proof of Concept
Establishing a POC is highly recommended to prove that the selected hardware and software 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

Related implementation projects that prepare the ground 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.

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 entity 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.

3.9.2 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 3.25**, 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, the communication processes is established with the analysis project to get 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 3.26**).

Delivery Time

During the delivery time phase, the delivery plan is developed (**figure 3.27**). 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, 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 3.28**).

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.

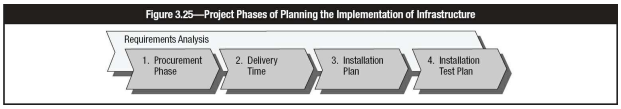


Figure 3.25—Project Phases of Planning the Implementation of Infrastructure

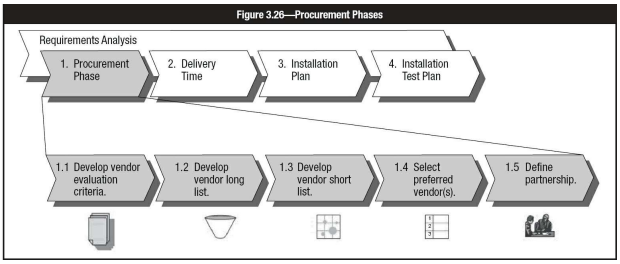


Figure 3.26—Procurement Phases

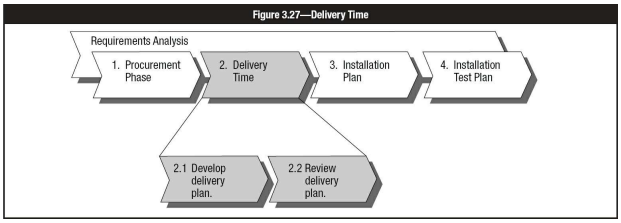


Figure 3.27—Delivery Time

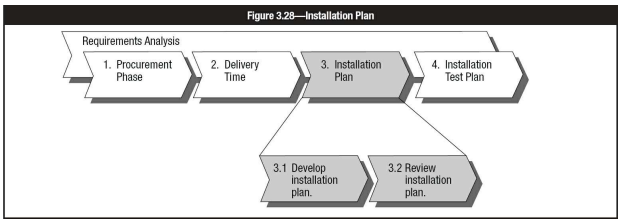


Figure 3.28—Installation Plan

Installation Test Plan
Based on the known dependencies of the installation plan, the test plan is developed (**figure 3.29**).

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.

3.9.3 CRITICAL SUCCESS FACTORS

Critical success factors of planning the implementation include:
- 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 quickly.
- Part one of the project (Analysis of Physical Architecture) must be completed, and the needed infrastructure decisions must be made.

3.9.4 HARDWARE ACQUISITION

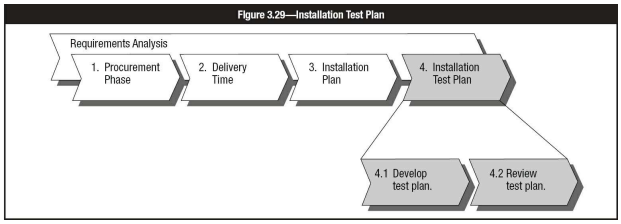
Selection of a computer hardware and software environment frequently requires the preparation of specifications for distribution to hardware/software (HW/SW) vendors and criteria for evaluating vendor proposals. The specifications are sometimes presented to vendors in the form of an ITT, also known as a 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 where 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, based on *ISO/IEC 15408:2009: Information technology—Security techniques—Evaluation criteria for IT security*; developing HW/SW requirements using common criteria–evaluated IT products provides a level of confidence (based in independent laboratory testing) that the security functionality of these IT products meet stated security specifications.
- 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)
- Adaptability requirements such as:
  - Hardware and software upgrade capabilities
  - Compatibility with existing hardware and software platforms
  - Changeover to other equipment capabilities

Figure 3.29—Installation Test Plan



- 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) hardware and software 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 "grey 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

When performing an audit of this area, the 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.

3.9.5 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

3.9.6 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.

3.10 INFORMATION SYSTEMS MAINTENANCE PRACTICES

System maintenance practices refer primarily to the process of managing change to application systems while maintaining the integrity of both the production hardware (e.g., network and related server management products) and application source and executable code.

After a system is moved into production, it seldom remains static. Change is expected in all systems regardless of whether they are vendor-supplied or internally developed. Reasons for change in normal operations include internal IT/business changes, new external regulations, changes in classification related to either sensitivity or criticality, audits, and adverse incidents such as intrusions and viruses.

To control the ongoing maintenance of the system, a standard process for performing and recording changes is necessary. This process, which will be an integral part of the organization's overall SDLC process, should include steps to ensure that the system changes are appropriate to the needs of the organization, appropriately authorized, documented, thoroughly tested and approved by management. The process typically is established in the design phase of the application when application system requirements are baselined.

3.10.1 CHANGE MANAGEMENT PROCESS OVERVIEW

The change management process begins with authorizing changes to occur. For this purpose, a methodology should exist for prioritizing and approving system change requests. Change requests are initiated from end users as well as operational staff and system development/maintenance staff. In any case, authorization needs to be obtained from appropriate levels of end-user and systems management (e.g., a change control group, configuration control boards). For acquired systems, a vendor may distribute periodic updates, patches or new release levels of the software. User and systems management should review such changes. Determination should be made as to whether the changes are appropriate for the organization or will negatively affect the existing system.

Users should convey system change requests to the system management using some type of formal correspondence such as a standard change request form,

memo or email message. At a minimum, the user request should include the requestor's name, date of request, date the change is needed, priority of the request, a thorough description of the change request, a description of any anticipated effects on other systems or programs, and fallback procedures in case the changes cause the failure of the system. The user could also provide a reason for the change, a cost justification analysis and the expected benefits of the change. In addition, the request should provide evidence that it has been reviewed and authorized by user management. A signature on the request form or memo typically provides this evidence.

Change requests should be in a format that ensures all changes are considered for action and allows the system management staff to easily track the status of the request. This is usually done by assigning a unique control number to each request and entering the change request information into a computerized system. This can also be performed manually. With detailed information regarding each request, management can identify those requests that have been completed and are still in progress or have not been addressed yet. Management can also use this information to help ensure that user requests are addressed in a timely manner. See **figure 3.30**.

All requests for changes and related information should have a security impact analysis performed to validate that change's impact on the system's overall security posture. The system maintenance staff as part of the system's permanent documentation should maintain the completed change documentation and supporting information.

Maintenance records of all program changes should exist either manually or automatically. Several library management software products provide this type of audit trail. The maintenance information usually consists of the programmer ID, time and date of change, project or request number associated with the change, and before and after images of the lines of code that were changed.

This process becomes even more important when the programmer who creates the program is also the operator. In this case, it is assumed that the IT department is either small or there are few applications being processed. Special change management procedures must be closely followed because SoD cannot be established in this environment and compensating controls are required. It requires user management to pay more attention to changes and upgrades made by the programmer, and proper authorization must be given to the programmer before putting any change into production. In lieu of the manual process of management approving changes before the programmer can submit them into production, management could have automated change control software installed to prevent unauthorized program changes. By doing this, the programmer is no longer responsible for migrating changes into production. The change control software becomes the operator that migrates programmer changes into production based on approval by management.

Programmers should not have write, modify or delete access to production data. Depending on the type of information in production, programmers may not even have read-only access (or access to customer credit card numbers, US Social Security numbers/national ID numbers or other sensitive information that may require added security).

Deploying Changes
After the end user is satisfied with the system test results and the adequacy of the system documentation, approval should be obtained from user management. Users should convey system change requests to the system management using some type of formal correspondence such as a standard change request form, memo or email message.

Documentation
To ensure the effective utilization and future maintenance of a system, it is important that all relevant system documentation be updated. Due to tight time constraints and limited resources, thorough updates to documentation are often neglected. Documentation requiring revision may consist of program and/or system flowcharts, program narratives, data dictionaries, entity relationship models, data flow diagrams (DFDs), operator run books and end-user procedural manuals. Keeping the internal coherence of all these items is a challenge; software configuration management packages can be a valuable tool.

Procedures should be in place to ensure that documentation stored offsite for disaster recovery purposes is also updated. This documentation is often overlooked and may be out of date.

Figure 3.30—Sample Change Request Form

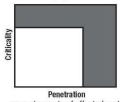
Request for Change (RFC) Document

1. Contents

This document shall ensure that, at a minimum, every major change will be applied to the customer's mission or business critical systems in a controlled environment. It shall support all affected parties in gaining a more reliable and resilient infrastructure. Thus the usage of this document is mandatory for all involved personnel and—during normal operations—has to be distributed in a timely manner to provide the setup of a Forward Schedule of Change. In the rare condition of emergency changes it must be belatedly completed for documentation purpose. **In either case there must always be a formal approval for the RFC covered in this document.**

2. Usage Guidance

The following figure shall illustrate the usage guideline for this document.



The area shaded in red symbolizes the mandatory usage of the formal RFC procedure depicted in this document.

If the proposed change has an impact either on critical/vital systems (criticality is high) or on a massive amount of systems (wide penetration) then it is—by definition—a major change and the RFC procedure has to be executed.

In any other case (area left blank on the figure) it is up to the requestor to choose a formal approach or to bypass the RFC procedure.

To follow best practices a list shall be compiled which itemizes all systems classified as either critical or vital so the requestor will know what to head for (criticality). Additionally a percentage rate of affected systems shall be published as a threshold so that it is clearly defined when to execute the RFC procedure (penetration).

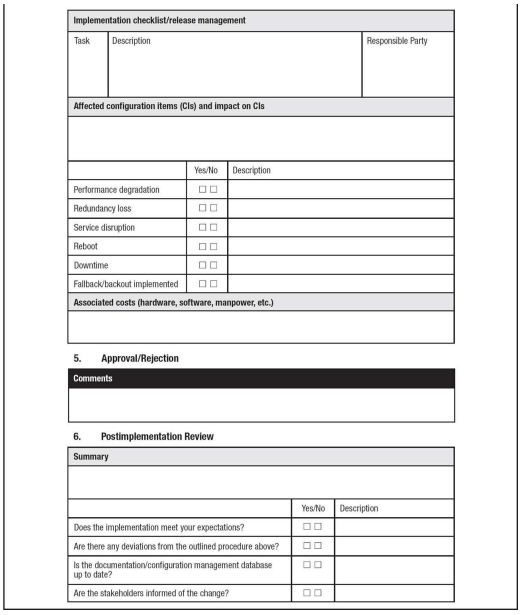
3. General RFC data

RFC title:				ID:	
Standard/emergency change:	☐ / ☐				
Issued by:		Intended recipient:			
Issued on:		Scheduled for:			
Checked by:		Checked on:		Status:	
Postimplementation review done by:		Postimplementation review done on:		Status:	

Has to be filled out by the requestor
Has to be filled out by the approver

4. Scope of Change

Abstract
Detailed description (please be as specific as you can)
Expected benefit



Implementation checklist/release management		
Task	Description	Responsible Party

Affected configuration items (CIs) and impact on CIs		

	Yes/No	Description
Performance degradation	☐ ☐	
Redundancy loss	☐ ☐	
Service disruption	☐ ☐	
Reboot	☐ ☐	
Downtime	☐ ☐	
Fallback/backout implemented	☐ ☐	

Associated costs (hardware, software, manpower, etc.)

5. Approval/Rejection

Comments

6. Postimplementation Review

Summary		
	Yes/No	Description
Does the implementation meet your expectations?	☐ ☐	
Are there any deviations from the outlined procedure above?	☐ ☐	
Is the documentation/configuration management database up to date?	☐ ☐	
Are the stakeholders informed of the change?	☐ ☐	

Testing Changed Programs

Changed programs should be tested and also eventually certified with the same discipline as newly developed systems to ensure that the changes perform the intended functions. In addition, if the risk analysis determines it is necessary, additional testing would be required to ensure:

- Existing functionality is not damaged by the change
- System performance is not degraded because of the change
- No security exposures have been created because of the change

Auditing Program Changes

In evaluating whether procedures for program changes are adequate, the IS auditor should ensure that controls are in place to protect production application programs from unauthorized changes. The control objectives are as follows:

- Access to program libraries should be restricted.
- Supervisory reviews should be conducted.
- Change requests should be approved and documented.
- Potential impact of changes should be assessed.
- The change request should be documented on a standard form, paying particular attention to the following:
  - The change specifications should be adequately described, a cost analysis developed and a target date established.
  - The change form should be signed by the user to designate approval.
  - The change form should be reviewed and approved by programming management.
  - The work should be assigned to an analyst, programmer and programming group leader for supervision.
- A sample of program changes made during the audit period should be selected and traced to the maintenance form to determine whether the changes are authorized, check that the form has appropriate approvals, and compare the date on the form with the date of production update for agreement.
- If an independent group updates the program changes in production, the IS auditor should determine before the update whether procedures exist to ensure possession of the change request form. (This is accomplished by watching the groups perform their jobs.)

Emergency Changes

There may be times when emergency changes are required to resolve system problems and enable critical “production job” processing to continue. Procedures should primarily exist in the application’s operations manual to ensure emergency fixes can be performed without compromising the integrity of the system. This typically involves the use of special logon IDs (i.e., emergency IDs) that grant a programmer/analyst temporary access to the production environment during these emergency situations. The use of emergency IDs should be logged and carefully monitored because their use grants someone powerful privileges. Emergency fixes should be completed using after-the-fact, follow-up procedures that ensure that all normal change management controls are retroactively applied. Changes completed in this fashion are held in a special emergency library from where they should be moved through the change management process into normal production libraries in an expeditious manner. IS auditors need to pay particular attention that emergency changes are handled in an appropriate and transparent manner.

Deploying Changes Back Into Production

Once user management has approved the change, the modified programs can be moved into the production environment. A group that is independent of computer programming should perform the migration of programs from test to production. Groups such as computer operations, QA or a designated change control group should perform this function.

To ensure that only authorized individuals have the ability to migrate programs into production, proper access restrictions must be in place. Such restrictions can be implemented through the use of OS security or an external security package.

Distributed systems, such as point-of-sale systems, offer an additional challenge in ensuring that changed programs are rolled out to all nodes. The rollout may be performed over a significant period of time to enable:

- Controls to be exercised over conversion of data
- Training of staff who will be using the changed software

- Support to be provided to users of the changed system
- Reduction of the risk associated with changing all nodes at the same time, and having to back them all out if something goes wrong

In view of the time it may take to roll out changes to the whole distributed system, controls must ensure that all nodes are eventually updated. If changes are made on a regular basis, it may be necessary to check regularly that all nodes are running the same versions of all software.

Change Exposures (Unauthorized Changes)
An unauthorized change to application system programs can occur for several reasons:
- The programmer has access to production libraries containing programs and data including object code.
- The user responsible for the application was not aware of the change (no user signed the maintenance change request approving the start of the work).
- A change request form and procedures were not formally established.
- The appropriate management official did not sign the change form approving the start of the work.
- The user did not sign the change form signifying acceptance before the change was updated into production.
- The changed source code was not properly reviewed by the appropriate programming personnel.
- The appropriate management official did not sign the change form approving the program for update to production.
- The programmer put in extra code for personal benefit (i.e., committed fraud).
- Changes received from the acquired software vendor were not tested or the vendor was allowed to load the changes directly into production/site. This happens in cases of distributed processing sites, such as POS, banking applications, ATM networks, etc.

3.10.2 CONFIGURATION MANAGEMENT

Because of the difficulties associated with exercising control over both system and programming maintenance activities, more and more organizations implement configuration management systems. In fact in many cases, regulatory requirements mandate these levels of control to provide a high degree of reliability and repeatability in all associated system processes (government systems, critical infrastructure, ICS, etc.). 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. From an audit perspective, effective use of this software provides important evidence of management's commitment to careful control over the maintenance process.

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 also 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 must support the concept of configuration management. 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. As part of the software configuration management task, the maintainer performs the following task steps:
1. Develop the configuration management plan.
2. Baseline the hardware, code, network physical and logical connections, ports protocols and services and associated engineering, operational, and administrative documents.
3. Analyze and report on the results of configuration control.
4. Develop the reports that provide configuration status information.
5. Develop release procedures.
6. Perform configuration control activities such as identification and recording of the request.
7. Update the configuration status accounting database.

In many cases, commercial software products will be used to automate the manual 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 by providing automated support for the following:
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 only be built from the baselined items. The baseline becomes the trusted recovery source for these systems and applications.

3.11 SYSTEM DEVELOPMENT TOOLS AND PRODUCTIVITY AIDS

System development tools and productivity aids include code generators, CASE applications and 4GL. These tools and aids are addressed in the following sections.

3.11.1 CODE GENERATORS

Code generators are tools, often incorporated with CASE products, that 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. The IS auditor should be aware of source code generated by such tools.

3.11.2 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 embrace two of these categories or all three of them.

CASE tools are available for mainframe, minicomputer and microcomputer environments. These tools can provide higher quality systems more quickly. CASE products enforce 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.

The 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.

The IS auditor should gain assurances 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 the 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 there needs to be a project methodology in place for CASE to be effective. The methodology 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.

The IS auditor may also become a user of CASE tools as several features facilitate the audit process. DFDs, which may be the product of upper and middle CASE tools, may be used as an alternative to other flowcharting techniques. IS auditors whose IS departments are moving into CASE are using CASE-generated documentation as part of the audit. Some are even experimenting with the use of CASE tools to create audit documentation. In addition, CASE tools can be used to develop interrogation software and embedded audit modules (EAMs). Repository reports should be used to gain an understanding of the system and to review controls over the development process.

3.11.3 FOURTH-GENERATION LANGUAGES

While a standard definition of a 4GL does not exist, the common characteristics of 4GLs are the following:

- **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, the 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 integrated development environment (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 (3GLs) such as COBOL and C. The application can be further tailored and customized. Data processing development personnel, not end users, use application generators.

3.12 PROCESS IMPROVEMENT PRACTICES

Business processes require improvements, which are accomplished with practices and techniques addressed in the following sections.

3.12.1 BUSINESS PROCESS REENGINEERING AND PROCESS CHANGE PROJECTS

The generic model shown in **figure 3.31** is a very basic description of a 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 (e.g., breakdown of subprocesses to activities) 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 (e.g., ISO 9001:2000).

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 shall be included in management reports for review under the plan-do-check-act (PDCA) cycle.

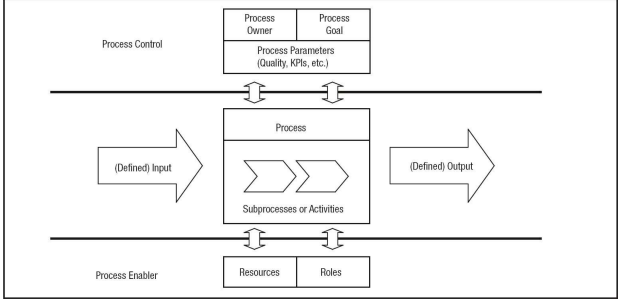
BPR is the process of responding to competitive and economic pressures, and customer demands to survive in the current business environment. This is usually done by automating system processes so that there are fewer manual interventions and manual controls. BPR achieved with the help of implementing an ERP system is often referred to as package-enabled reengineering (PER). Advantages of BPR are usually experienced where 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:
- 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:
- 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, JIT 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

Figure 3.31—Generic Process Model



- 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 the baseline processes 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 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. The IS auditor's task is to identify the existing key controls and evaluate the impact of removing these controls. If the controls are key preventive controls, the IS auditor must ensure that management is aware of the removal of the control and management is willing to accept the potential material risk of not having that preventive control.

BPR Methods and Techniques
Applying BPR methods and techniques to a process creates an immediate environment for change and provides consistency of results.

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 a world-class "reference" 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, SWOT (strengths, weaknesses, opportunities and threats) analysis
- Investment decisions, company takeovers, mergers
- Product or process design or redesign/reengineering
- BPR

The steps listed below are followed generally by the benchmarking team in a benchmarking exercise:
1. **Plan**—In the planning stage, critical processes are identified for the benchmarking exercise. The benchmarking team should identify the critical processes and understand how they are measured, the kinds of data that are needed and how the data need to be collected.
2. **Research**—The team should collect baseline data about the processes of its own organization before collecting these 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, consultants, etc. 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.
   - Depending on the aims of the exercise and the resulting scenario above, the next steps will 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 will be 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.

BPR Audit and Evaluation
When reviewing an organization's business process change (reengineering) efforts, IS auditors must determine whether:
- The organization's change efforts are consistent with the overall culture and strategic plan of the organization.
- The reengineering team is making an effort 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.

The IS auditor would also provide a statement of assurance or conclusion with respect to the objectives of the audit.

3.12.2 ISO/IEC 25010:2011

ISO/IEC 25010:2011 is an international standard to assess the quality of software products. It provides the definition of the characteristics and associated quality evaluation process to be used when specifying the requirements for, and evaluating the quality of, software products throughout their life cycle. Attributes evaluated include:
- **Functionality**—The set of attributes that bears on the existence of a set of functions and their specified properties. The functions are those that satisfy stated or implied needs.
- **Reliability**—The set of attributes that bears on the capability of software to maintain its level of performance under stated conditions for a stated period of time.
- **Usability**—The set of attributes that bears on the effort needed for use and on the individual assessment of such use by a stated or implied set of users.
- **Efficiency**—The set of attributes that bears on the relationship between the level of performance of the software and the amount of resources used under

stated conditions.
- **Maintainability**—The set of attributes that bears on the effort needed to make specified modifications.
- **Portability**—The set of attributes that bears on the ability of software to be transferred from one environment to another.

3.12.3 CAPABILITY MATURITY MODEL INTEGRATION

Following the release and successful adoption of capability maturity model (CMM) for Software, other models were developed for disciplines such as systems engineering, integrated product development, etc. The Capability Maturity Model Integration (CMMI) was conceived as a means of combining the various models into a set of integrated models. CMMI also describes five levels of maturity, although the descriptions of what constitutes each level differ from those used in the original CMM. CMMI is considered less directly aligned with the traditional waterfall approach toward development and better aligned with contemporary software development practices including:
- Iterative development
- Early definition of architecture
- Model-based design notation
- Component-based development
- Demonstration-based assessment of intermediate development products
- Use of scalable, configurable processes

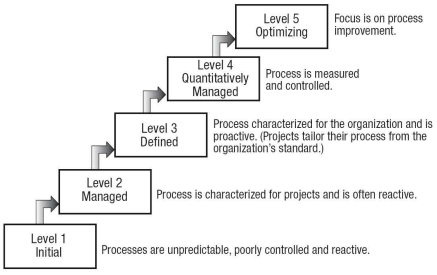
Maturity models, such as CMMI, are useful to evaluate management of a computer center, the development function management process, and implement and measure the IT change management process.

See **figure 3.32** for characteristics of the maturity levels.

3.12.4 ISO/IEC 330XX SERIES

ISO/IEC 3300xx is a series of standards that provide guidance on process assessment. It supersedes and extends parts of the ISO/IEC 15504 series of standards. ISO/IEC 33002:2015 lists requirements for effective process assessment. ISO/IEC 33004:2015 provides requirements for process reference, assessment and maturity models. ISO/IEC 33020:2015 describes a process measurement framework for the assessment of process capability. Other ISO/IEC 330xx documents further outline the relationship of this series to the ISO/IEC 15504 series and cover other topics such as process improvement.

Figure 3.32—Characteristics of the Maturity Levels (CMMI)



Source: Adapted from Godfrey, Sally; *Using CMMI for Improvement at GSFC, Systems Engineering Seminar*, NASA, USA, 2004; based on Software Engineering Institute; *CMMI® for Development*, Version 1.3, USA, 2010

3.13 APPLICATION CONTROLS

Application controls are controls over 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 system integrity, that applicable system functions operate as intended, and that information contained by the system is relevant, reliable, secure and available when needed.

The IS auditor's tasks include the following:
- Identifying the significant application components and the flow of transactions through the system and gaining a detailed understanding of the application by reviewing the available documentation and interviewing appropriate personnel
- Identifying the application control strengths, and evaluating the impact of the control weaknesses
- Developing a testing strategy
- Testing the controls to ensure their functionality and effectiveness by applying appropriate audit procedures
- Evaluating the control environment by analyzing the test results and other audit evidence to determine that control objectives were achieved
- Considering 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

3.13.1 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 that these transactions are only processed once. 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 at the time 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**—Necessary to ensure that access authorization cannot be compromised through use of another individual's authorized data access. Individual passwords also provide accountability for data changes. (See chapter 5 Protection of Information Assets, for more information.)
- **Terminal or client workstation identification**—Used to limit 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**—The forms used to record 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.

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 taken into account. 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.

Batch header forms are a data preparation control. All input forms should be clearly identified with the application name and transaction codes. Where possible, preprinted and prenumbered forms, with transaction identification codes and other constant data items, are recommended. This would help ensure that all pertinent data have been recorded on the input forms and can reduce data recording/entry errors.

Types of batch controls include:
- **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 total in a batch agrees with the total calculated by the system. Hash total is the total of non-value numeric fields in the batch (like 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 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**—These registers enable recording of batch totals and subsequent comparison with system reported totals.
- **Control accounts**—Control account use is performed 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**—Computer agreement with batch totals is performed through the input of batch header details that record the batch totals; the system compares these 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 batches 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 include:
- **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**—Written evidence of user, data entry and data control procedures
- **Error correction procedures**—These include:
  - Logging of errors
  - Timely corrections
  - Upstream resubmission
  - Approval of corrections
  - Suspense file
  - Error file
  - Validity of corrections
- **Anticipation**—The user or control group anticipates 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

3.13.2 PROCESSING PROCEDURES AND CONTROLS

Processing procedures and controls are meant to ensure the reliability of application program processing. IS auditors need to 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 3.33** describes various types of data validation edits.

Processing Controls

Processing controls are meant to ensure the completeness and accuracy of accumulated data. They would 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:
- **Manual recalculations**—A sample of transactions may be recalculated manually to ensure that processing is accomplishing the anticipated task.
- **Editing**—An edit check is 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**—Run-to-run totals provide the ability to verify 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 can be used to detect and initiate 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**—Application programs can verify 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**—An edit check can provide assurance, 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.

Figure 3.33—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.

- **Reconciliation of file totals**—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**—An exception report is 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 3.34**.

Contents of data files, or indeed 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.

Figure 3.34—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.

- **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 the application represent the management design of controls on how a business process (procurement or sales or payroll) should be run. While the applications contain 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 real-world production operations, in some situations, entities may have to carry out direct updates to 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 is entered or processed. In such cases, management may take a decision to catch-up on the backlog by directly updating the transactions in the database (back end). Therefore, the 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 the quality of application controls is only as good as the quality of controls around direct back-end data fixes, in any entity.

3.13.3 OUTPUT CONTROLS

Output controls are meant to 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. The IS auditor should validate that the reports are accurate and correct representation of the source data

Example 3

In a trial balance report (where the debit side and credit side is expected to total, if the application had processed transactions accurately), the programmer was found to have coded the report in such a manner that the report simply reproduced the credit total under the debit column to force the tie out of the debit and credit totals.

The IS auditor needs to apply an assessment approach in validating reports depending on the situation (more evaluation when the entity has undergone a system change or evaluating customized reports as against standard reports of a widely used application).

- **Report distribution**—Output reports should be distributed according to authorized distribution parameters, which may be automated or manual. Operations personnel should verify that output 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 adequately to ensure that no unauthorized access can occur. Reports that are distributed electronically through the computer system also need to be considered. Logical access to these reports should also be controlled carefully and subject to authorization. When distributed manually, assurance should be provided that sensitive reports are properly distributed. Such assurance should include the recipient signing a log as evidence of receipt of output (i.e., manual nonrepudiation).
- **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.

The 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)

3.13.4 BUSINESS PROCESS CONTROL ASSURANCE

In an integrated application environment, controls are embedded and designed into the application that supports the processes. Business process control assurance involves evaluating controls at the process and activity level. These controls may be a combination of management, programmed and manual controls. In addition to evaluating general controls that affect the processes, business process owner-specific controls—such as establishing proper security and SoD, periodic review and approval of access, and application controls within the business process—are evaluated.

Specific matters to consider in the business process control assurance are:
- Process and data flow mapping
- Process controls
- Assessing business risks within the process
- Benchmarking with best practices
- Roles and responsibilities
- Activities and tasks
- Data restrictions

3.14 AUDITING APPLICATION CONTROLS

The IS auditor's tasks include the following:
- Identifying the significant application components and the flow of information through the system, and gaining a detailed understanding of the application by reviewing the available documentation and interviewing appropriate personnel. Developing a data flow diagram can help visualize the flow of information.
- Understanding and evaluation of interfaces, including APIs, in case the application connects with other applications
- Identifying the application control strengths and evaluating the impact of the control weaknesses to develop a testing strategy by analyzing the accumulated information
- Reviewing application system documentation to provide an understanding of the functionality of the application. In many cases—mainly in large systems or packaged software—it is not feasible to review the whole application documentation. Thus, a selective review should be performed. If an application is vendor supplied, technical and user manuals should be reviewed. Any changes to applications should be documented properly.

The following documentation should be reviewed to gain an understanding of an application's development:
- **System development methodology documents**—These documents include cost-benefit analysis and user requirements.
- **Functional design specifications**—This document provides a detailed explanation of the application. An understanding of key control points should be noted during review of the design specifications.
- **Program changes**—Documentation of any program change should be available for review. Any change should provide evidence of authorization and should be cross-referenced to source code.
- **User manuals**—A review of the user manuals provides the foundation for understanding how the user is utilizing the application. Often control weaknesses can be noted from the review of this document.
- **Technical reference documentation**—This documentation includes any vendor-supplied technical manuals for purchased applications in addition to any in-house documentation. Access rules and logic usually are included in these documents.

3.14.1 FLOW OF TRANSACTIONS THROUGH THE SYSTEM

A transaction flowchart provides information regarding key processing controls. Points where transactions are entered, processed and posted should be reviewed for control weaknesses.

3.14.2 RISK ASSESSMENT MODEL TO ANALYZE APPLICATION CONTROLS

Risk assessment, as discussed in chapter 1, provides information relating to the inherent risk of an application.

A risk assessment model can be based on many factors, which may include a combination of the following:
- The quality of internal controls
- Economic conditions (impacts on organizational resource availability, capacity and capability)
- 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

3.14.3 OBSERVING AND TESTING USER PERFORMING PROCEDURES

Some of the user procedures that should be observed and tested include:

- **Segregation of duties**—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**—Evidence of input authorization can be achieved via written authorization on input documents or with the use of unique passwords. One may test this 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**—This is performed to verify 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**—This is completed in the form of reports that provide evidence of appropriate review, 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**—Critical output reports should be produced and maintained in a secure area and distributed 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**—Access control tables provide information regarding access levels by individuals. 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**—These 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**—These indicate 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.

3.14.4 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 will 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.

3.14.5 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, and hence, each transaction only accesses 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.

This type of testing is vital in today's vast array of online Internet-accessible, multiuser DBMSs.

3.14.6 TEST APPLICATION SYSTEMS

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 3.35**.

To facilitate the evaluation of application system tests, an IS auditor may also want to use generalized audit software (GAS), also known as computer-assisted audit tools (CAATs). 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, GAS can be used to perform certain application control tests, such as parallel simulation, in comparing expected outcomes to live data.

Figure 3.35—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

3.14.7 CONTINUOUS ONLINE AUDITING

Continuous online auditing is becoming increasingly important in the e-business world because it provides a method for the IS auditor to collect evidence on system reliability while normal processing takes place. The approach allows IS auditors to monitor the operation of such a system on a continuous basis and gather selective audit evidence through the computer. If the selective information collected by the computer technique is not deemed serious or material enough to warrant immediate action, the information is stored in separate audit files for verification by the IS auditor at a later time. The continuous audit approach cuts down on needless paperwork and leads to the conduct of an essentially paperless audit. In such a setting, an IS auditor can report directly through the microcomputer on significant errors or other irregularities that may require immediate management action. This approach reduces audit cost and time.

Continuous audit techniques are important IS audit tools, particularly when they are used in time-sharing environments that process a large number of transactions but leave a scarce paper trail. By permitting IS auditors to evaluate operating controls on a continuous basis without disrupting the organization's usual operations, continuous audit techniques improve the security of a system. When a system is misused by someone withdrawing money from an inoperative account, a continuous audit technique will report this withdrawal in a timely fashion to the IS auditor. Thus the time lag between the misuse of the system and the detection of that misuse is reduced. The realization that failures, improper manipulation and lack of controls will be detected on a timely basis by the use of continuous audit procedures gives IS auditors and management greater confidence in a system's reliability.

Continuous audit very often relies on calls to GAS/CAAT services.

3.14.8 ONLINE AUDITING TECHNIQUES

There are five types of automated evaluation techniques applicable to continuous online auditing:

1. **Systems Control Audit Review File and Embedded Audit Modules (SCARF/EAM)**—The use of this technique involves embedding specially written audit software in the organization's host application system so the application systems are monitored on a selective basis.
2. **Snapshots**—This technique involves taking what might be termed pictures of the processing path that a transaction follows, from the input to the output stage. With the use of this technique, transactions are tagged by applying identifiers to input data and recording selected information about what occurs for the auditor's subsequent review.
3. **Audit hooks**—This technique involves embedding hooks in application systems to function as red flags and to induce IS security and auditors to act before an error or irregularity gets out of hand.
4. **Integrated test facility (ITF)**—In this technique, dummy entities are set up and included in an auditee's production files. The IS auditor can make the system either process live transactions or test transactions during regular processing runs and have these transactions update the records of the dummy entity. The operator enters the test transactions simultaneously with the live transactions that are entered for processing. The auditor then compares the output with the data that have been independently calculated to verify the correctness of the computer-processed data.
5. **Continuous and intermittent simulation (CIS)**—During a process run of a transaction, the computer system simulates the instruction execution of the application. As each transaction is entered, the simulator decides whether the transaction meets certain predetermined criteria and, if so, audits the transaction. If not, the simulator waits until it encounters the next transaction that meets the criteria.

In **figure 3.36**, the relative advantages and disadvantages of the various concurrent audit tools are presented.

The use of each of the continuous audit techniques has advantages and disadvantages. Their selection and implementation depends, to a large extent, on the complexity of an organization's computer systems and applications, and the IS auditor's ability to understand and evaluate the system with and without the use of continuous audit techniques. In addition, IS auditors must recognize that continuous audit techniques are not a cure for all control problems and that the use of these techniques provides only limited assurance that the information processing systems examined are operating as they were intended to function.

3.15 AUDITING SYSTEMS DEVELOPMENT, ACQUISITION AND MAINTENANCE

The IS auditor's tasks in system development, acquisition and maintenance may take place once the project is finished or during the project itself. Most tasks in the following list cover both scenarios and the IS auditor is expected to determine which task applies. 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 risks 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 risks 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. Also review and evaluate the application system audit trails to ensure that documented controls are in place to address all security, edit and processing controls. Audit trails are tracking mechanisms that can help IS auditors ensure program change accountability. 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 quality assurance activities
- Identify and test existing controls to determine the adequacy of production library security to ensure the integrity of the production resources.
- Participate in postimplementation reviews.
- 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.

3.15.1 PROJECT MANAGEMENT

Throughout the project management process the 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, the 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. The IS auditor should also assess the project team's ability to produce key deliverables by the promised dates.

Typically, the 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

Figure 3.36—Concurrent Audit Tools—Advantages and Disadvantages

	SCARF/EAM	Snapshots	Audit Hooks	ITF	CIS
Complexity	Very high	Medium	Low	High	Medium
Useful when:	Regular processing cannot be interrupted.	An audit trail is required.	Only select transactions or processes need to be examined.	It is not beneficial to use test data.	Transactions meeting certain criteria need to be examined.

- 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

3.15.2 FEASIBILITY STUDY

The IS auditor should perform the following functions:
- Review the documentation produced in this phase for reasonableness.
- Determine whether all cost justifications/benefits are verifiable and showing the anticipated costs and benefits to be realized.
- Identify and determine the criticality of the need.
- 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.

3.15.3 REQUIREMENTS DEFINITION

The IS auditor should perform the following functions:
- Obtain the detailed requirements definition document and verify its accuracy 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.

3.15.4 SOFTWARE ACQUISITION PROCESS

The IS auditor should perform the following functions:
- 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.

3.15.5 DETAILED DESIGN AND DEVELOPMENT

The 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 the input, processing and output controls designed into the system for appropriateness.
- 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 EAMs were coded into the appropriate programs.

3.15.6 TESTING

Testing is crucial in determining the user requirements have been validated, the system is performing as anticipated and internal controls work as intended. Therefore, it is essential that the 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 user acceptance testing 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.

3.15.7 IMPLEMENTATION PHASE

This phase is initiated only after a successful testing phase. The system should be installed according to the organization's change control procedures. The 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 that all recent updates from the testing phase have been incorporated.
- Verify all data conversion to ensure that they are correct and complete before implementing the system in production.

3.15.8 POSTIMPLEMENTATION REVIEW

After the new system has stabilized in the production environment, a postimplementation review should be performed. Prior to this review, it is important that sufficient time be allowed for the system to stabilize in production. In this way, any significant problems will have had a chance to surface.

The IS auditor should perform the following functions:
- Determine if the system's objectives and requirements were achieved. During the postimplementation 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, use this module 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.

3.15.9 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, the 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.

Additionally, the IS auditor should review the overall change management process for possible improvements in acknowledgement, response time, response effectiveness and user satisfaction with the process.

3.16 CASE STUDIES

The following case studies are included as a learning tool to reinforce the concepts introduced in this chapter.

3.16.1 CASE STUDY A

A major retailer asked the IS auditor to review their readiness for complying with credit card company requirements for protecting cardholder information. The IS auditor subsequently learned the following information. The retailer uses wireless POS registers that connect to application servers located at each store. These registers use wired equivalent protection (WEP) encryption. The application server, usually located in the middle of the store's customer service area, forwards all sales data over a frame relay network to database servers located at the retailer's corporate headquarters, and using strong encryption over an Internet virtual private network (VPN) to the credit card processor for approval of the sale. Corporate databases are located on a protected screened subset of the corporate local area network. Additionally, weekly aggregate sales data by product line is copied from the corporate databases to magnetic media and mailed to a third party for analysis of buying patterns. It was noted that the retailer's database software has not been patched in over two years. This is because vendor support for the database package was dropped due to management's plans to eventually upgrade to a new ERP system.

CASE STUDY A QUESTIONS	
A1.	Which of the following would present the **MOST** significant risk to the retailer? A. Wireless POS registers use WEP encryption. B. Databases patches are severely out-of-date. C. Credit cardholder information is sent over the Internet. D. Aggregate sales data are mailed to a third party.
A2.	Based on the case study, which of the following controls would be the **MOST** important to implement? A. Store application servers should be located in a secure area. B. POS registers should use two-factor authentication. C. Wireless access points should use MAC address filtering. D. Aggregate sales data sent offsite should be encrypted.
See answers and explanations to the case study questions at the end of the chapter (page 237).	

3.16.2 CASE STUDY B

A large industrial concern has begun a complex IT project, with ERP, to replace the main component systems of its accounting and project control departments. 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. In the meanwhile, detailed design and programming of the next phase begins. After a

period of initial adjustment, the pilot users start experiencing serious difficulties. In spite of positive test results, already stabilized functionalities began to have intermittent problems; transactions hang during execution, and more and more frequently, project data are corrupted in the database. Additional problems show up—errors already corrected started occurring again and functional modifications already tested tend to present other errors. The project, already late, is now in a critical situation. The IS auditor, after collecting the evidence, requests an immediate meeting with the head of the project steering committee to communicate findings and suggest actions capable to improve the situation.

CASE STUDY B QUESTIONS	
B1.	The IS auditor should indicate to the head of the project steering committee that: A. the observed project problems are a classic example of loss of control of project activities and loose discipline in following procedures and methodologies. A new project leader should be appointed. B. relays due to an underestimation of project efforts have led to failures in the versioning and modification control procedures. New programming and system resources must be added to solve the root problem. C. the problems are due to excessive system modifications after each delivery phase. The procedure for control of modifications must be tightened and made more selective. D. the nature of initial problems is such as to lead to doubts regarding the adequacy and reliability of the platform. An immediate technical review of the hardware and software platform (parameters, configuration) is necessary.
B2.	In order to contribute more directly to solve the situation, the IS auditor should: A. research the problems further to identify root causes and define appropriate countermeasures. B. review the validity of the functional project specifications as the basis for an improved software baselining definition. C. propose to be included in the project team as a consultant for the quality control of deliverables. D. contact the project leader and discuss the project plans and recommend redefining the delivery schedule using the PERT methodology.
See answers and explanations to the case study questions at the end of the chapter (page 237).	

3.17 ANSWERS TO CASE STUDY QUESTIONS

ANSWERS TO CASE STUDY A QUESTIONS

A1. **A** 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. With regard to the unpatched database servers, since they are located on a screened subnet, this would mitigate the risk to the organization. Sending credit cardholder data over the Internet would be less of a risk because strong encryption is being utilized. Because the sales data being sent to the third party are aggregate data, no cardholder information should be included.

A2. **A** Locating application servers in an unsecured area creates a significant risk to the retailer because cardholder data may be retained on the server. Additionally, physical access to the server could potentially allow programs or devices to be installed that would capture sensitive cardholder information. Two-factor authentication for POS registers is not as critical. In the case of MAC address filtering, this can be spoofed. Because sales data are aggregate, no cardholder information should be at risk when the data are sent to a third party.

ANSWERS TO CASE STUDY B QUESTIONS

B1. **D** The IS auditor has only found symptoms but not the root causes of the severe problems, so attributing them to limited skills of the project leader is inappropriate. The sequence of negative events is to show that a fundamental problem is causing serious technical difficulties and, as a consequence of delays, negatively impacting the versioning and change control procedures. It is necessary, however, to ascertain immediately the nature of the suspect problems so if such a problem exists and is not found, it could severely affect the final result or kill the project. Also, the added technical programming resources are an inappropriate remedy because the development methodology is RAD, which is based on using a small group of skilled and experienced technical resources. The initial effects of the problem (intermittent block of transactions, corrupted data) lead to the suspicion that a problem might exist in the setup and configuration of the technical hardware/software environment or platform.

B2. **A** 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. Functional project specifications should be executed by users and systems analysts, and not by the auditor. 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. 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 environmen