



Federal Financial Institutions Examination Council

FFIEC

Development and
Acquisition

D&A

APRIL 2004

IT EXAMINATION

HANDBOOK

TABLE OF CONTENTS

INTRODUCTION	1
Examination Objectives	1
Standards	2
Accounting for Software Costs	2
Information Security	2
PROJECT MANAGEMENT	3
System Development Life Cycle	4
Alternative Development Methodologies	4
Roles and Responsibilities	5
Project Plans	6
Project Management Standards	8
Project Planning Standards	8
Configuration Management Standards	9
Quality Assurance Standards	9
Risk Management Standards	10
Testing Standards	10
Documentation Standards	11
Project Management Tools	11
Gantt Charts	12
Project Evaluation Review Techniques	12
Groupware	12
Project Management Effectiveness	13
Capability Maturity Model®	13
International Organization for Standardization	14
DEVELOPMENT	15
Development Standards	16
Systems Development Life Cycle	16
Initiation Phase	17
Planning Phase	19
Design Phase	21
Development Phase	24
Testing Phase	29
Implementation Phase	30
Project Evaluation	31
Maintenance Phase	31
Disposal Phase	32

Large-Scale Integrated Systems	33
Software Development Techniques.....	33
Object-Oriented Programming.....	33
Computer-Aided Software Engineering.....	34
Rapid Application Development	35
Databases.....	37
Database Management Systems	38
ACQUISITION	39
Acquisition Standards.....	40
Acquisition Project Guidance.....	41
Escrowed Documentation.....	43
Software Development Contracts and Licensing Agreements	44
Overview.....	44
Software Licenses - General	45
Software Licenses and Copyright Violations	45
Software Development Specifications and Performance Standards.....	46
Documentation, Modification, Updates, and Conversion	47
Bankruptcy.....	47
Regulatory Requirements.....	47
Payments.....	47
Representations and Warranties.....	48
Dispute Resolution	48
Agreement Modifications	48
Vendor Liability Limitations.....	49
Security.....	49
Subcontracting and Multiple Vendor Relationships.....	50
Restrictions on Adverse Comments	50
MAINTENANCE	51
Major Modifications.....	52
Routine Modifications	52
Emergency Modifications	54
Patch Management	55
Library Controls	55
Conversions.....	56
Utility Controls.....	57
Documentation Maintenance.....	57
APPENDIX A: EXAMINATION PROCEDURES	59
APPENDIX B: GLOSSARY	68

INTRODUCTION

The “Development and Acquisition Booklet” is one in a series of booklets updating the 1996 Federal Financial Institutions Examination Council (FFIEC) Information Systems Handbook (FFIEC IS Handbook). The booklet, which rescinds Chapter 12 of the 1996 FFIEC IS Handbook provides examiners and financial institutions guidance for identifying and controlling development and acquisition risks.¹

Development and acquisition is defined as “an organization’s ability to identify, acquire, install, and maintain appropriate information technology systems.”² The process includes the internal development of software applications or systems and the purchase of hardware, software, or services from third parties.³

The development, acquisition, and maintenance processes include numerous risks. Effective project management influences operational risks (also referred to as transactional risks). These risks include the possibility of loss resulting from inadequate processes, personnel, or systems. Losses can result from errors; fraud; or an inability to deliver products or services, maintain a competitive position, or manage information. Refer to the FFIEC Information Technology Examination Handbook (IT Handbook’s) “Management Booklet” for additional information.

The Development and Acquisition Booklet describes common project management activities and emphasizes the benefits using well-structured project management techniques. The booklet details general project management standards, procedures, and controls and discusses various development, acquisition, and maintenance project risks. Action summaries highlight the primary considerations within each section. Examiners should use the summaries to identify primary issues within each section, but should be aware the summaries are not substitutes for reading the entire document.

EXAMINATION OBJECTIVES

The objectives of reviewing development, acquisition, and maintenance activities are to identify weaknesses or risks that could negatively impact an organization, to identify entities whose condition or performance requires special supervisory attention, and to subsequently effect corrective action.

¹ This booklet uses the terms “organization” and “financial institution” to describe insured banks, thrifts, and credit unions, as well as the service providers that furnish technology-related services to such entities.

² Federal Financial Institutions Examination Council Uniform Rating System for Information Technology

³ The acquisition activities discussed in this booklet cover the acquisition of software products. Refer to the IT Handbook’s “Outsourcing Technology Services Booklet” for guidance relating to the acquisition of third-party services and outsourced software development projects.

Examiners should conduct risk-focused reviews that assess the overall effectiveness of an organization's project management standards, procedures, and controls. Examiners should not expect organizations to employ elaborate project management techniques in all situations. However, organizations should employ project management standards, procedures, and controls commensurate with characteristics and risks of their development, acquisition, and maintenance projects.

STANDARDS

The critical importance technology plays in financial institutions dictates the use of appropriate development, acquisition, and maintenance standards. Standards do not guarantee that organizations will appropriately develop, acquire, and maintain technology systems. However, standards do enhance management's control over projects, thereby decreasing project risks. Well-defined standards help ensure systems are obtained in an efficient manner, operate in a secure and reliable environment, and meet organizational and end-user needs. Therefore organizations that routinely complete projects should establish comprehensive standards, policies, and procedures that meet project and organizational needs and reduce project risks.

ACCOUNTING FOR SOFTWARE COSTS

Organizations must correctly account for costs associated with the acquisition and development of software for internal use. The American Institute of Certified Public Accountants' Statement of Position (SOP) 98-1 requires organizations to capitalize or expense various costs associated with obtaining and developing internally used software. Management should become familiar with SOP 98-1 and other applicable accounting standards and discuss specific capitalization and expense issues with its accountants.

INFORMATION SECURITY

Information security is a critical part of internally and externally developed software. Financial institutions should consider information security requirements and incorporate automated controls into internally developed programs, or ensure the controls are incorporated into acquired software, before the software is implemented. For additional details, please refer to the Handbook's "Information Security Booklet" and additional industry standards such as Security Considerations in the Information System Development Life Cycle published by the National Institute of Standards and Technology.

PROJECT MANAGEMENT

Action Summary

Financial institutions should establish appropriate development, acquisition, and maintenance project management methodologies. The methodologies should match a project's characteristics and risks and include appropriate:

- f* Project plans;
- f* Definitions of project requirements and expectations;
- f* Project management standards and procedures;
- f* Quality assurance and risk management standards and procedures;
- f* Definitions of project roles and responsibilities;
- f* Involvement by all affected parties; and
- f* Project communication techniques.

Project management in its basic form involves planning and completing a task. Technology-related tasks include ongoing operational activities and one-time projects. A project's impact on operations must be a key consideration when assessing development, acquisition, and maintenance activities.

Detailed project plans, clearly defined expectations, experienced project managers, realistic budgets, and effective communication significantly enhance an organization's ability to manage projects successfully. Effectively managed projects often result in late deliveries, cost overruns, poor quality applications.

Inferior applications can result in underdeveloped, insecure, or unreliable systems. Retrofitting functional, security, or automated-control features into applications is expensive, time consuming, and often results in less effective features. Therefore organizations must manage projects carefully to ensure they are products that meet organizational needs on time and within budget.

Financial institutions use various methods to manage technology projects. The systems development life cycle (SDLC) is the primary project management methodology described in this booklet. The SDLC is used for illustrative purposes because it provides a systematic way to describe the numerous tasks associated with software development projects. Organizations may employ SDLC model or alternative methodology when managing any project, including software development, or hardware, software, or service acquisition projects. Regardless of the method used, it should be tailored to match a project's characteristics and risks. Boards, or board-designated committees, should

formally approve project methodologies, and management should approve and document significant deviations from approved procedures.

SYSTEM DEVELOPMENT LIFE CYCLE

Structured project management techniques (such as an SDLC) enhance management's control over projects by dividing complex tasks into manageable sections. Segmenting projects into logical control points (phases) allows managers to review project phases for successful completion before allocating resources to subsequent phases.

The number of phases within a project's life cycle is based on the characteristics of a project and the employed project management methodology. A five-step process may only include broadly defined phases such as prepare, acquire, implement, and maintain. Typical software development projects include initiation, planning, design, development, testing, implementation, and maintenance phases. Some organizations include a final, disposal phase in their project life cycles.

The activities completed within each project phase are also based on the project type and project management methodology. All projects should follow well-structured plans that clearly define the requirements of each project phase.

ALTERNATIVE DEVELOPMENT METHODOLOGIES

The SDLC provides a logical approach to managing a sequential series of tasks. However, a drawback to using a traditional SDLC is that project risks may not be adequately controlled if tasks are completed in a strictly sequential manner. For example, using a traditional SDLC methodology, users define functional requirements and pass them to system designers. Designers complete the designs and pass them to programmers. If programmers subsequently discover improved ways to provide the functional requirements, the designers must redo their work. However, if programmers are involved in the planning and design phases, they may be able to identify improvements earlier in the process. Therefore, to enhance the effectiveness of project activities, organizations should employ methodologies that involve all parties in each project phase.

Development techniques such as spiral, iterative, and modified SDLC methodologies address many of the shortcomings of a traditional SDLC. Full descriptions of the newer methodologies are beyond the scope of this document. However, examiners should be aware that the newer methodologies are more focused and involve the completion of project phases in repetitive (iterative) cycles. Iteration enhances a project manager's ability to efficiently address the requirements of each party (end users, security administrators, designers, developers, system technicians, etc.) throughout a project's life cycle. Iteration also allows project managers to complete, review, and revise phase activities until they produce satisfactory results (phase deliverables).

ROLES AND RESPONSIBILITIES

The size and complexity of a project dictate the required number and qualifications of project personnel. Duties may overlap in larger organizations or lower-risk projects; however, all projects should include appropriate segregation of duties or compensating controls.

Primary roles and responsibilities include:

- f* Corporate Management – Corporate managers are responsible for approving major projects and ensuring projects support, not drive, business objectives.
- f* Senior Management – Senior managers are responsible for approving and promoting projects with their authority and ensuring adequate resources are available to complete projects.
- f* Technology Steering Committee – Technology steering committees are responsible for establishing and approving major project deliverables and coordinating interdepartmental activities. The committees often include the project manager, a board member, and executives from all organizational departments. Large organizations often establish project management offices to coordinate multiple projects.
- f* Project Manager – Project managers are responsible for ensuring projects support business objectives, project goals and expectations are clearly defined, and project tasks are identified, scheduled, and completed. Project managers are also responsible for monitoring and reporting a project's status to senior management.
- f* Project Sponsor – Project sponsors are responsible for developing support within user departments, defining deliverables, and providing end users for testing purposes. Project sponsors often provide financial resources to a project.
- f* Technology Department – The technology department is responsible for maintaining the technology resources used by project teams and assisting in the testing and implementation phases. Department members should assist in defining the scope of a project by identifying database and network resources and constraints.
- f* Quality Assurance – Quality assurance personnel are responsible for validating project assumptions and ensuring the quality of phase deliverables. Quality assurance personnel should be independent of the development process and use predefined standards and procedures to assess deliverables throughout project life cycles.
- f* User Departments – User departments assist project managers, designers, and programmers in defining and testing functional requirements (system features). End-user involvement throughout a project is critical to ensuring accurate definitions and adequate tests.

Large projects may include a subject matter expert or data analyst responsible for communicating user information and functional requirements to project teams.

- f Auditors – Auditors assist user departments, project managers, and system designers in identifying system control requirements and testing the controls during development and after implementation.
- f Security Managers – Security managers assist user departments, project managers, and system designers in identifying security requirements and testing the features during development and after implementation.

PROJECT PLANS

Planning activities are the most critical aspect of project management due to the high number of interrelated project tasks. Poor planning often contributes to projects failing to meet expectations. Therefore, examiners must carefully assess the adequacy of an organization's project planning activities. Examiners should focus their assessments on management's ability to develop and employ project plans that are appropriately tailored to match a project's characteristics and risks.

The initiation phase is when a project request is submitted. Requests should justify the rationale for a project (present a business case), identify desired system features and, to the extent possible, define the overall project scope. The scope of a project includes ancillary items such as information requirements, network interfaces, and hardware components that support and interact with a requested product. Management should determine if the business case justifies the project scope (by considering issues such as tangible and intangible benefits, estimated costs, projected return-on-investment, etc.) and decide if the project is feasible. If management approves a request, the scope documentation serves as the basis for developing the project plan.

Project plans refine the scope documentation by further identifying the specific activities and resources required to complete a project. Plans should address project work activities and project management activities. Work activity planning involves organizing project teams, scheduling tasks, allocating resources, etc. Project management planning involves establishing project and risk management procedures, documenting project objectives and assumptions, defining documentation and reporting standards, etc.

Formal project plans should include:

- f Project Overviews – Project overviews detail the background of a project and explain general project objectives and strategies.
- f Roles and Responsibilities – The identification of key personnel and description of primary responsibilities enhance each team members' understanding of project assignments and reporting requirements.

- f* Communication Procedures – Standardized communication and reporting procedures enhance the exchange of information between project personnel, particularly on large projects.
- f* Defined Deliverables – Clearly filed project requirements and acceptance criteria are necessary to ensure management and employees understand expectations.
- f* Standards – Project management, change control, and quality assurance standards increase the likelihood of project success.
- f* Control Requirements – Automated control and security features that are designed into applications gain a project's life cycle enhance the features' effectiveness.
- f* Quality Assurance Plan – Quality assurance plans help ensure projects and products meet organizational standards and expectations.
- f* Risk Management – Risk identification, assessment, and control procedures increase the likelihood of project success.
- f* Configuration Management – Configuration management plans (which describe methods for controlling and documenting changes to established project plans, service requirements, and hardware and software configurations) enhance project and maintenance efficiencies.
- f* Documentation – Identification of the type and level of documentation that team members must produce throughout each project phase can help to increase a project's effectiveness and enhances maintenance capabilities.
- f* Budget – Preliminary budgets that estimate project costs enhance management's ability to assess a project's feasibility. Monitoring budgets throughout a project helps management assess and control expenditures.
- f* Scheduling – Project phase and activity schedules increase a project's effectiveness.
- f* Testing – Test plans and schedules enhance test efficiencies and effectiveness.
- f* Staff Development – Training plans and schedules enhance training efficiencies and effectiveness.

PROJECT MANAGEMENT STANDARDS

Organizations should establish project management standards. Institutions that routinely complete multiple projects should establish project management offices to coordinate project activities.

Standards should be in place to address general project activities such as project request, review, and approval processes, project management methodology selections, and project reporting and documentation requirements. Standards should also address the specific requirements of individual projects. For example, standards relating to software development projects should include, among other things, application design, programming, and testing requirements.

Project management standards should be commensurate with organizational and project characteristics and risks. The standards should be sufficiently detailed to ensure team members can identify project objectives and expectations. Clearly defined expectations are a prerequisite for successfully completing projects and obtaining buy-in throughout an organization. The standards should require representatives from all departments involved in, or affected by, a project to assist in defining functional requirements and project deliverables.

Organizations that need to coordinate multiple projects should establish standards for coordinating and managing the projects from an organization-wide perspective. The standards should include procedures for project prioritizing, resource coordination, progress reporting, stakeholder project resolution, etc.

PROJECT PLANNING STANDARDS

Organizations should establish appropriate project planning standards. The standards should require management to develop project plans that are detailed in proportion to a project's characteristics and risks. Management should develop well-defined plans for all projects.

Project plans should describe existing systems, benefits and weaknesses, explain project goals, and identify user, information, system, and network requirements. Such explanations and descriptions enhance team members' abilities to understand project objectives and develop systems that meet organizational needs. Project plans should identify quality assurance procedures; risk management procedures, including security features which will be needed; testing procedures; and documentation procedures. Additionally, project plans should detail cost, staffing, resource, and training requirements.

⁴ As noted in the IT Handbook's "Management Booklet", standards are criteria mandated by management to ensure corporate conformity with policy, government regulations, and acceptable levels of control.

CONFIGURATION MANAGEMENT STANDARDS

Organizations should establish configuration management standards. Configuration management involves controlling project component changes in order to minimize project disruptions and ensure original objectives are addressed. Configuration management standards should require the identification of baseline configurations (original versions) of hardware, software services, documentation, and project management plans. Standards should also be in place to ensure all changes to baseline versions are evaluated, approved, documented, and disseminated. Refer to the “Maintenance” section for additional configuration management details.

QUALITY ASSURANCE STANDARDS

Quality assurance is a critical part of well-managed development and acquisition projects. Comprehensive quality assurance, risk management, and testing standards provide the best means to manage project risks and ensure software includes expected functionality, security, and operability. Quality assurance procedures should be applied to internally and externally developed programs.

Management should establish quality assurance standards that address:

- f Commitment – Successful projects require commitment from all involved parties. Senior management should adequately support and promote projects throughout an organization to enhance organizational acceptance of a project. End users should assist in defining and testing functional requirements and project teams should efficiently complete tasks. All parties should clearly define their expectations and effectively communicate throughout a project. Management’s failure to implement or support quality assurance programs decreases an organization’s ability to detect project weaknesses and programming errors quickly. The later weaknesses and errors are detected, the more difficult and costly they are to correct.
- f Completeness – Each phase within a project life cycle includes procedures to follow and items to deliver. Therefore, quality assurance programs should parallel phases of the life cycle. For example, the project initiation phase includes the presentation of a business case, the request for desired functional requirements, and the identification of interconnected system components. Quality assurance personnel should verify justification for a project, the necessity of requested functional features, and the accuracy of system connections before projects move into the project planning phase. Audit and compliance employees should assist quality assurance personnel verify compliance with internal and external project requirements.

ensure that appropriate standards exist to protect the confidentiality of that data. Management can use test data generators, which are software applications that generate representative testing data based upon predefined parameters, to develop appropriate testing data. Numerous automated applications are also available that test program logic, functional operability, and network interoperability.

DOCUMENTATION STANDARDS

Organizations should establish appropriate documentation standards. Documentation consists of detailed descriptions and explanations of technology applications, systems, and procedures. Documentation enhances a user's ability to use, review, or modify the applications, systems, and procedures. Management should maintain documentation for all technology resources, including nontechnical policy and procedural guidance, and technical information such as hardware and software configurations, and system and application source codes. The quality and quantity of the documentation should be commensurate with the characteristics and risks of the associated resource. For example, high risk applications should be more formally documented than applications that are considered low risk by the organization.

Development and acquisition project documentation should include project requests, feasibility studies, project plans, testing plans, etc. System documentation, which focuses on system analysis and design, should include system concept narratives, data flow charts, and database specifications. Application documentation should include application descriptions, programming flowcharts, and operations and user instructions.

PROJECT MANAGEMENT TOOLS

Project managers use various tools to schedule and monitor project tasks and estimate project costs and completion dates. Experienced project managers can effectively use internally developed spreadsheets for small or noncomplex projects. However, in most cases, managers should develop or purchase more sophisticated software applications when managing large, complex projects.

The Gantt chart (described below) illustrates a simple project management tool. The tools (applications) used to manage larger projects are much more elaborate and managers rely upon them heavily to make project decisions. For example, managers can use the tools at key decision points or milestones to generate reports identifying whether a project is meeting time and budget estimates. Managers may also use the applications to insert report comments describing any issues or problems affecting a project. Management should implement appropriate access controls and backup procedures to ensure the security and reliability of these critical project management tools.

GANTT CHARTS

Managers can use Gantt charts to monitor project tasks. The charts identify project milestones on the vertical axis and time estimates on the horizontal axis. The charts provide an easy way to observe the overall status of a project, but do not effectively identify interrelated tasks or problems.

Gantt Chart	Jan	Feb	Mar	Apr
Initiation Phase	xxxxxxxx			
Planning Phase	===	=====		
Design Phase		----	-----	
Development Phase			----	-----
Scheduled: ----- Started: ===== Completed: xxxxx				

PROJECT EVALUATION REVIEW TECHNIQUES

Managers can use Project Evaluation Review Technique (PERT) charts in conjunction with Gantt charts to identify and administer interrelated tasks. PERT charts present project tasks, interrelationships, and time estimates as network diagrams. The diagrams present clear connections between project phases or key milestones.

GROUPWARE

Project management tools also include groupware, sometimes referred to as collaboration software. Groupware consists of software applications that facilitate communication and data exchange between working groups. The products combine email, calendar, conferencing, and document-management functions to enhance productivity. Typically the products are employed in local- or wide-area networks because internal documents are easier to update than those distributed in HTML formats over the Internet. However, Internet based products that take advantage of more robust protocols such as XML are becoming increasingly available.

Note: When assessing the effectiveness of project management tools, examiners should verify the effectiveness and accuracy of the input process and evaluate the complexity of project plans. Entering poorly defined requirements or unrealistic expectations into planning tools defeats the purpose of trying to develop realistic goals. Overly complex plans may limit effectiveness because they can be difficult to understand and maintain.

PROJECT MANAGEMENT EFFECTIVENESS

There are several methods for enhancing the effectiveness of an organization's project management skills. Typically the methods involve training project personnel and developing structured management techniques. The following sections include two examples for illustrative purposes. The inclusion of these items in this booklet is not an endorsement or all-inclusive representation of the methods.

CAPABILITY MATURITY MODEL®

The Carnegie Mellon University Software Engineering Institute (SEI) developed the Capability Maturity Model for Software® (Model) to assist organizations in the assessment and improvement of their project management process. The Model categorizes an organization's capability to develop software within one of five "maturity" levels. The Model suggests an organization can improve its development skills (move into a higher category) by implementing the techniques defined within a category (and within any lower categories, excluding the Initial category). The categories and defining characteristics are:

- Initial – Organizations with Initial capabilities use ad hoc development techniques and limited standardized procedures.
- Repeatable – Organizations with Repeatable capabilities use fundamental project planning, scheduling, and monitoring procedures.
- Defined – Organizations with Defined capabilities use approved, formalized management and development standards and procedures, customized to meet specific project requirements, for all projects.
- Managed – Organizations with Managed capabilities measure, understand, and control development procedures and product quality.
- Optimizing – Organizations with Optimized capabilities use effective communication techniques and innovative ideas and technologies to obtain ongoing improvements in processes and products.

The main areas SEI encourages organizations to implement or improve to mature their development capabilities include:

- Repeatable – Basic project management controls such as requirements, configuration, and subcontractor management controls; software project planning, tracking, and oversight procedures; and software quality assurance programs;
- Defined – Project and organizational issues such as process focus, training programs, integrated software management, software product engineering, intergroup coordination, and peer reviews;

- **Managed** – Quantitative issues such as software quality and process management; and
- **Optimizing** – Organizational and project issues such as defect prevention, technology change controls, and process change management.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION

The International Organization for Standardization (ISO) is comprised of standards institutes from around the world. The nongovernmental organization includes approximately 150 members with representatives from private and public sectors. The organization's primary goal is to facilitate the development and coordination of product and service standards that are designed to enhance private sector trade and governmental legislation development.

The ISO framework includes highly specific standards that are applied to particular products and processes and generic management standards that can be applied to any organization. It is important to note that the International Standards Organization does not certify compliance with ISO standards. Private companies and, in some countries, governmental bodies evaluate organizations to determine their compliance with ISO standards.

The ISO 9000 series of standards includes ISO 9001, which addresses management practices relating to design, development, production, installation, and servicing activities. The generic 9001 standards focus on manufacturing activities. However, the organization published ISO 9000-3 guidelines to assist project managers in applying the 9001 standards in software development environments.

DEVELOPMENT

Action Summary

Financial institutions should establish appropriate systems and application development methodologies. The methodologies should match a project's characteristics and risks and require appropriate:

- Project plans;
- Definitions of project expectations;
- Project standards and procedures;
- Definitions of project phase deliverables, including assurance that deliverables will meet any applicable legal and regulatory requirements;
- Development of security, audit, and automated-control features;
- Quality assurance, risk management, and testing standards and procedures;
- Involvement by all affected parties; and
- Project communication techniques.

Development projects involve the creation of software applications or integrated application systems. Software development projects are completed in-house, through outsourcing, or by a combined approach. Organizations typically manage development projects using systematic methodologies that divide large, complex tasks into smaller, more easily managed segments or phases.

Traditionally, many organizations used the systems development life cycle method to assist in developing software for use in mainframe operating environments. The SDLC provided a satisfactory method to manage the projects because the functional and security requirements of the software were limited. Functional requirements were primarily limited to transaction processing and output reporting. Security requirements were limited because of the closed environment in which mainframes operated. Typically, few individuals had access to the mainframes. Therefore, physical restrictions over mainframe terminals and logical controls over program and data libraries provided most of a system's security.

Client/server systems provide significantly more users increased access to systems and data. Therefore, the need to develop software with greater functionality and stronger internal controls contributed to the development of alternative, risk-focused software development techniques.

Alternative development techniques (such as spiral, iterative, and modified SDLC methodologies) involve the completion of project activities in repetitive (iterative) cycles. The techniques reduce project risks by ensuring the requirements of each participant (end

users, auditors, security administrators, designers, developers, system technicians, etc.) are thoroughly considered during each project phase. Involving all parties during each project phase reduces the risk that organizations will not identify problems until late in a project's life cycle. The newer methodologies often employ prototyping or modeling techniques during initial project phases. Prototyping enhances user's ability to visualize how systems will look and work after the systems are installed.

DEVELOPMENT STANDARDS

Organizations should establish development standards that, at a minimum, address project management, system control, and quality assurance issues. Project management standards should address issues such as project management methodologies, risk management procedures, and project approval authorities. System control standards should address items such as an application's functional, security, and automated control features. Quality assurance standards should address issues such as the validation of project assumptions, adherence to project standards, and testing of a product's performance.

Development standards should include procedures for managing changes during the development process. "Scope creep" is a common problem associated with software development projects. It occurs when developers receive requests to add or modify a program's features while the program is being developed. Although the addition or modification of functional, security, or control features may be appropriate, uncontrolled changes disrupt the development process. Establishing change approval procedures and cut-off dates (after which requested changes are deferred to subsequent versions) assist organizations manage change during the development process.

Development standards should also include procedures for managing internally developed spreadsheets and database reports. Financial institutions often rely on the spreadsheets and reports to make important budgeting and asset/liability decisions, but fail to implement adequate testing, documentation, and change-control procedures. Management's reliance on the spreadsheets and reports should dictate the formality of their development procedures, change controls, and backup techniques.

SYSTEMS DEVELOPMENT LIFE CYCLE

The systems development life cycle is a project management technique that divides complex projects into smaller, more easily managed segments or phases. Segmenting projects allows managers to verify the successful completion of project phases before allocating resources to subsequent phases.

Software development projects typically include initiation, planning, design, development, testing, implementation, and maintenance phases. However, the phases may be divided differently depending on the organization involved. For example, initial project activities might be designated as request, requirements-definition, and planning

phases, or initiation, concept-development, and planning phases. End users of the system under development should be involved in reviewing the output of each phase to ensure the system is being built to deliver the needed functionality.

Note: Examiners should focus their assessments of development, acquisition, and maintenance activities on the effectiveness of an organization's project management techniques. Reviews should be centered on ensuring the depth, quality, and sophistication of a project management technique are commensurate with the characteristics and risks of the project under review.

INITIATION PHASE

Careful oversight is required to ensure projects support strategic business objectives and resources are effectively implemented into an organization's enterprise architecture. The initiation phase begins when an opportunity to add, improve, or correct a system is identified and formally requested through the presentation of a business case. The business case should, at a minimum, describe a proposal's purpose, identify expected benefits, and explain how the proposed system supports one of the organization's business strategies. The business case should also identify alternative solutions and detail as many informational, functional, and network requirements as possible.

The presentation of a business case provides a point for managers to reject a proposal before they allocate resources to a formal feasibility study. When evaluating software development requests (and during subsequent feasibility and design analysis), management should consider input from all affected parties. Management should also closely evaluate the necessity of each requested functional requirement. A single software feature approved during the initiation phase can require several design documents and hundreds of lines of code. It can also increase testing, documentation, and support requirements. Therefore, the initial rejection of unnecessary features can significantly reduce the resources required to complete a project.

If provisional approval to initiate a project is obtained, the request documentation serves as a starting point to conduct a more thorough feasibility study. Completing a feasibility study requires management to verify the accuracy of the preliminary assumptions and identify resource requirements in greater detail.

Primary issues organizations should consider when compiling feasibility study support documentation include:

- Business Considerations:
 - Strategic business and technology goals and objectives;
 - Expected benefits measured against the value of current technology;
 - Potential organizational changes regarding facilities or the addition/reduction of end users, technicians, or managers;

- Budget, scheduling, or personnel constraints; and
- Potential business, regulatory, or legal issues that could impact the feasibility of the project.
- Functional Requirements:
 - End-user functional requirements;
 - Internal control and information security requirements;
 - Operating, database, and backup system requirements (type, capacity, performance);
 - Connectivity requirements (stand-alone, Local Area Network, Wide Area Network, external);
 - Network support requirements (number of potential users; type, volume, and frequency of data transfers); and
 - Interface requirements (internal or external applications).
- Project Factors:
 - Project management methodology;
 - Risk management methodology;
 - Estimated completion dates of projects and major project phases; and
 - Estimated costs of projects and major project phases.
- Cost/Benefit Analysis:
 - Expected useful life of the proposed product;
 - Alternative solutions (buy vs. build);
 - Nonrecurring project costs (personnel, hardware, software, and overhead);
 - Recurring operational costs (personnel, maintenance, telecommunications, and overhead);
 - Tangible benefits (increased revenues, decreased costs, return-on-investments); and
 - Intangible benefits (improved public opinions or more useful information).

The feasibility support documentation should be compiled and submitted for senior management or board study. The feasibility study document should provide an overview of the proposed project and identify expected costs and benefits in terms of economic, technical, and operational feasibility. The document should also describe alternative solutions and include a recommendation for approval or rejection. The document should be reviewed and signed off on by all affected parties. If approved, management should use the feasibility study and support documentation to begin the planning phase.

PLANNING PHASE

The planning phase is the most critical step in completing development, acquisition, and maintenance projects. Careful planning, particularly in the early stages of a project, is necessary to coordinate activities and manage project risks effectively. The depth and formality of project plans should be commensurate with the characteristics and risks of a given project.

Project plans refine the information gathered during the initiation phase by further identifying the specific activities and resources required to complete a project. A critical part of a project manager's job is to coordinate discussions between user, audit, security, design, development, and network personnel to identify and document as many functional, security, and network requirements as possible.

Primary items organizations should address in formal project plans include:

- **Project Overview** – Project overviews provide an outline of the project plan. Overviews should identify the project, project sponsors, and project managers; and should describe project goals, background information, and development strategies.
- **Roles and Responsibilities** – Project plans should define the primary responsibilities of key personnel, including project sponsors, managers, and team members. Additionally, project plans should identify the responsibilities of third-party vendors and internal audit, security, and network personnel.
- **Communication** – Defined communication techniques enhance project efficiencies. Therefore, management should establish procedures for gathering and disseminating information. Standard report forms, defined reporting requirements, and established meeting schedules facilitate project communications.
- **Defined Deliverables** – Clearly defined expectations are a prerequisite for successfully completing projects. Representatives from all departments involved in, or affected by, a project should assist in defining realistic project objectives, accurate informational, functional, and interface requirements, and objective acceptance criteria.

Management should establish acceptance criteria for each project phase. Management should also establish appropriate review and approval procedures to ensure project teams complete all phase requirements before moving into subsequent phases.

- **Control Requirements** – An essential part of the planning process involves designing and building automated control and security features into applications. Identifying all required features and exactly where they should be placed is not always possible during initial project phases. However, management should consider

security and control issues throughout a project's life cycle and include those features in applications as soon as possible during a project's life cycle.

- Risk Management – Managing risks is an important part of the project planning process. Organizations should establish procedures to ensure managers appropriately assess, monitor, and manage internal and external risks throughout a project's life cycle. The procedures should include risk acceptance, mitigation, and/or transfer strategies.

External risks include issues such as vendor failures, regulatory changes, and natural disasters. Internal risks include items that affect budgets, such as inaccurate cost forecasting or changing functional requirements; scheduling difficulties, such as unexpected personnel changes or inaccurate development assumptions; and work flow challenges, such as weak communication or inexperienced project managers.

- Change Management – Personnel often request the addition or modification of functional requirements during software development projects. Although the addition or modification of requirements may be appropriate, standards should be in place to control changes in order to minimize disruptions to the development process. Project managers should establish cut-off dates after which they defer requested changes to subsequent versions. Additionally, representatives from the same departments involved in establishing requirements should be involved in evaluating and approving proposed changes. Large, complex, or mission-critical projects should include formal change management procedures.
- Standards – Project plans should reference applicable standards relating to project oversight activities, system controls, and quality assurance. Oversight standards should address project methodology selections, approval authorities, and risk management procedures. System controls standards should address functional, security, and automated-control requirements. Quality assurance standards should address the validity of project assumptions, adherence to project standards, and testing of a product's overall performance. Management should review, approve, and document deviations from established standards.
- Documentation – Project plans should identify the type and level of documentation personnel must produce during each project phase. For instance, personnel should document project objectives, system requirements, and development strategies during the initiation phase. The documentation should be revised as needed throughout the project. For example, preliminary user, operator, and maintenance manuals created during the design phase should be revised during

the development and testing phases, and finalized during the implementation phase.

- Scheduling – Management should identify and schedule major project phases and the tasks to be completed within each phase. Due to the uncertainties involved with estimating project requirements, management should build flexibility into project schedules. However, the amount of flexibility built into schedules should decline as projects progress and requirements become more defined.
- Budget – Managers should develop initial budget estimations of overall project costs so they can determine if projects are feasible. Managers should monitor the budgets throughout a project and adjust them if needed; however, they should retain a baseline budget for post-project analysis. In addition to budgeting personnel expenses and outsourced activities, it is important to include the costs associated with project overhead such as office space, hardware, and software used during the project.
- Testing – Management should develop testing plans that identify testing requirements and schedule testing procedures throughout the initial phases of a project. End users, designers, developers, and system technicians may be involved in the testing process.
- Staff Development – Management should develop training plans that identify training requirements and schedule training procedures to ensure employees are able to use and maintain an application after implementation.

DESIGN PHASE

The design phase involves converting the informational, functional, and network requirements identified during the initiation and planning phases into unified design specifications that developers use to script programs during the development phase. Program designs are constructed in various ways. Using a top-down approach, designers first identify and link major program components and interfaces, then expand design layouts as they identify and link smaller subsystems and connections. Using a bottom-up approach, designers first identify and link minor program components and interfaces, then expand design layouts as they identify and link larger systems and connections.

Contemporary design techniques often use prototyping tools that build mock-up designs of items such as application screens, database layouts, and system architectures. End users, designers, developers, database managers, and network administrators should review and refine the prototyped designs in an iterative process until they agree on an acceptable design. Audit, security, and quality assurance personnel should be involved in the review and approval process.

Management should be particularly diligent when using prototyping tools to develop automated controls. Prototyping can enhance an organization's ability to design, test, and establish controls. However, employees may be inclined to resist adding additional controls, even though they are needed, after the initial designs are established.

Designers should carefully document completed designs. Detailed documentation enhances a programmer's ability to develop programs and modify them after they are placed in production. The documentation also helps management ensure final programs are consistent with original goals and specifications.

Organizations should create initial testing, conversion, implementation, and training plans during the design phase. Additionally, they should draft user, operator, and maintenance manuals.

Application Control Standards

Application controls include policies and procedures associated with user activities and the automated controls designed into applications. Controls should be in place to address both batch and on-line environments. Standards should address procedures to ensure management appropriately approves and control overrides. Refer to the *IT Handbook's* "Operations Booklet" for details relating to operational controls.

Designing appropriate security, audit, and automated controls into applications is a challenging task. Often, because of the complexity of data flows, program logic, client/server connections, and network interfaces, organizations cannot identify the exact type and placement of the features until interrelated functions are identified in the design and development phases. However, the security, integrity, and reliability of an application is enhanced if management considers security, audit, and automated control features at the onset of a project and includes them as soon as possible in application and system designs. Adding controls late in the development process or when applications are in production is more expensive, time consuming, and usually results in less effective controls.

Standards should be in place to ensure end users, network administrators, auditors, and security personnel are appropriately involved during initial project phases. Their involvement enhances a project manager's ability to define and incorporate security, audit, and control requirements. The same groups should be involved throughout a project's life cycle to assist in refining and testing the features as projects progress.

Application control standards enhance the security, integrity, and reliability of automated systems by ensuring input, processed, and output information is authorized, accurate, complete, and secure. Controls are usually categorized as preventative, detective, or corrective. Preventative controls are designed to prevent unauthorized or invalid data entries. Detective controls help identify unauthorized or invalid entries. Corrective controls assist in recovering from unwanted occurrences.

Input Controls

Automated input controls help ensure employees accurately input information, systems properly record input, and systems either reject, or accept and record, input errors for later review and correction. Examples of automated input controls include:

- Check Digits – Check digits are numbers produced by mathematical calculations performed on input data such as account numbers. The calculation confirms the accuracy of input by verifying the calculated number against other data in the input data, typically the final digit.
- Completeness Checks – Completeness checks confirm that blank fields are not input and that cumulative input matches control totals.
- Duplication Checks – Duplication checks confirm that duplicate information is not input.
- Limit Checks – Limit checks confirm that a value does not exceed predefined limits.
- Range Checks – Range checks confirm that a value is within a predefined range of parameters.
- Reasonableness Checks – Reasonableness checks confirm that a value meets predefined criteria.
- Sequence Checks – Sequence checks confirm that a value is sequentially input or processed.
- Validity Checks – Validity checks confirm that a value conforms to valid input criteria.

Processing Controls

Automated processing controls help ensure systems accurately process and record information and either reject, or process and record, errors for later review and correction. Processing includes merging files, modifying data, updating master files, and performing file maintenance. Examples of automated processing controls include:

- Batch Controls – Batch controls verify processed run totals against input control totals. Batches are verified against various items such as total dollars, items, or documents processed.
- Error Reporting – Error reports identify items or batches that include errors. Items or batches with errors are withheld from processing, posted to a suspense account until corrected, or processed and flagged for later correction.
- Transaction Logs – Users verify logged transactions against source documents. Administrators use transaction logs to track errors, user actions, resource usage, and unauthorized access.

- Run-to-Run Totals – Run-to-run totals compiled during input, processing, and output stages are verified against each other.
- Sequence Checks – Sequence checks identify or reject missing or duplicate entries.
- Interim Files – Operators revert to automatically created interim files to validate the accuracy, validity, and completeness of processed data.
- Backup Files – Operators revert to automatically created master-file backups if transaction processing corrupts the master file.

Output Controls

Automated output controls help ensure systems securely maintain and properly distribute processed information. Examples of automated output controls include:

- Batch Logs – Batch logs record batch totals. Recipients of distributed output verify the output against processed batch log totals.
- Distribution Controls – Distribution controls help ensure output is only distributed to authorized individuals. Automated distribution lists and access restrictions on information stored electronically or spooled to printers are examples of distribution controls.
- Destruction Controls – Destruction controls help ensure electronically distributed and stored information is destroyed appropriately by overwriting outdated information or demagnetizing (degaussing) disks and tapes. Refer to the *IT Handbook's* “Information Security Booklet” for more information on disposal of media.

DEVELOPMENT PHASE

The development phase involves converting design specifications into executable programs. Effective development standards include requirements that programmers and other project participants discuss design specifications before programming begins. The procedures help ensure programmers clearly understand program designs and functional requirements.

Programmers use various techniques to develop computer programs. The large transaction-oriented programs associated with financial institutions have traditionally been developed using procedural programming techniques. Procedural programming involves the line-by-line scripting of logical instructions that are combined to form a program.

Primary procedural programming activities include the creation and testing of source code and the refinement and finalization of test plans. Typically, individual programmers write and review (desk test) program modules or components, which are small routines that perform a particular task within an application. Completed components are integrated with other components and reviewed, often by a group of programmers, to ensure the components properly interact. The process continues as component groups are progressively integrated and as interfaces between component groups and other systems are tested.

Advancements in programming techniques include the concept of "object-oriented programming". Object-oriented programming centers on the development of reusable program routines (modules) and the classification of data types (numbers, letters, dollars, etc.) and data structures (records, files, tables, etc.). Linking pre-scripted module objects to predefined data-class objects reduces development times and makes programs easier to modify. Refer to the "Software Development Techniques" section for additional information on object-oriented programming.

Organizations should complete testing plans during the development phase. Additionally, they should update conversion, implementation, and training plans and user, operator, and maintenance manuals.

Development Standards

Development standards should be in place to address the responsibilities of application and system programmers. Application programmers are responsible for developing and maintaining end-user applications. System programmers are responsible for developing and maintaining internal and open-source⁵ operating system programs that link application programs to system software and subsequently to hardware. Managers should thoroughly understand development and production environments to ensure they appropriately assign programmer responsibilities.

Development standards should prohibit a programmer's access to data, programs, utilities, and systems outside their individual responsibilities. Library controls can be used to manage access to, and the movement of programs between, development, testing, and production environments. Management should also establish standards requiring programmers to document completed programs and test results thoroughly. Appropriate documentation enhances a programmer's ability to correct programming errors and modify production programs.

⁵ Programs with licenses that generally provide users the freedom to run, study, modify, and redistribute either the original or modified program without paying royalties to previous developers. For additional details, refer to the Escrowed Documentation information in the Acquisition section.

Coding standards, which address issues such as the selection of programming languages and tools, the layout or format of scripted code, and the naming conventions of code routines and program libraries, are outside the scope of this document. However, standardized, yet flexible, coding standards enhance an organization's ability to decrease coding defects and increase the security, reliability, and maintainability of application programs. Examiners should evaluate an organization's coding standards and related code review procedures.

Library Controls

Libraries are collections of stored documentation, programs, and data. Program libraries include reusable program routines or modules stored in source or object code formats. Program libraries allow programmers to access frequently used routines and add them to programs without having to rewrite the code. Dynamic link libraries include executable code programs that can automatically run as part of larger applications.

Library controls should include:

- Automated Password Controls – Management should establish logical access controls for all libraries or objects within libraries. Establishing controls on individual objects within libraries can create security administration burdens. However, if similar objects (executable and non-executable routines, test and production data, etc.) are grouped into separate libraries, access can be granted at library levels.
- Automated Library Applications – When feasible, management should implement automated library programs, which are available from equipment manufacturers and software vendors. The programs can restrict access at library or object levels and produce reports that identify who accessed a library and what, if any, changes were made.

Version Controls

Library controls facilitate software version controls. Version controls provide a means to systematically retain chronological copies of revised programs and program documentation.

Development version control systems, sometimes referred to as concurrent version systems, assist organizations in tracking different versions of source code during development. The systems do not simply identify and store multiple versions of source code files. They maintain one file and identify and store only changed code. When a user requests a particular version, the system recreates that version. Concurrent version systems facilitate the quick identification of programming errors. For example, if programmers install a revised program on a test server and discover programming errors, they only have to review the changed code to identify the error.

Software Documentation

Organizations should maintain detailed documentation for each application and application system in production. Thorough documentation enhances an organization's ability to understand functional, security, and control features and improves its ability to use and maintain the software. The documentation should contain detailed application descriptions, programming documentation, and operating instructions. Standards should be in place that identify the type and format of required documentation such as system narratives, flowcharts, and any special system coding, internal controls, or file layouts not identified within individual application documentation.

Management should maintain documentation for internally developed programs and externally acquired products. In the case of acquired software, management should ensure (either through an internal review or third-party certification) prior to purchase, that an acquired product's documentation meets their organization's minimum documentation standards. For additional information regarding acquired software distinctions (open/closed code) refer to the "Escrowed Documentation" discussion in the "Acquisition" section.

Examiners should consider access and change controls when assessing documentation activities. Change controls help ensure organizations appropriately approve, test, and record software modifications. Access controls help ensure individuals only have access to sections of documentation directly related to their job functions.

System documentation should include:

- System Descriptions – System descriptions provide narrative explanations of operating environments and the interrelated input, processing, and output functions of integrated application systems.
- System Documentation – System documentation includes system flowcharts and models that identify the source and type of input information, processing and control actions (automated and manual), and the nature and location of output information.
- System File Layouts – System file layouts describe collections of related records generated by individual processing applications. For example, personnel may need system file layouts to describe interim files, such as sorted deposit transaction files, in order to further define master file processing requirements.

Application documentation should include:

- Application Descriptions – Application descriptions provide narrative explanations of the purpose of an application and provide overviews of data input, processing, and output functions.
- Layouts – Layouts represent the format of stored and displayed information such as database layouts, screen displays, and hardcopy information.

- **Program Documentation** – Program documentation details specific data input, processing, and output instructions, and should include documentation on system security. Program listings/source code and related narrative comments are the most basic items in program documentation and consist of technical programming scripts and non-technical descriptions of the scripts. It is important that developers update the listings and comment documentation when they modify programs. Many software development tools are available that automatically create source listings and narrative descriptions.

Traditionally, designers and developers have used flowcharts to present pictorial views of the sequencing of procedural programs such as COBOL and Assembler. Flowcharts provide a practical way to illustrate complex programs and routines. Flowcharting software is available that can automatically chart programs or enable programmers to chart programs dynamically without the need to draw them manually.

Programming techniques, such as object-oriented programming, have contributed to the use of dynamic flowcharting products. Maintaining detailed documentation of object-oriented code is particularly important because a primary benefit of the programming technique is the reuse of program objects.

- **Naming Conventions** – Naming conventions are a critical part of program documentation. Software programs are comprised of many lines of code, usually arranged hierarchically into small groups of code (modules, subroutines, or components), that perform individual functions within an application. Programmers should name and document the modules and any related subroutines, databases, or programs that interact with an application. Standardized naming conventions allow programmers to link subroutines into a unified program efficiently and facilitate technicians' and programmers' ability to understand and modify programs.
- **Operator Instructions** – Organizations should establish operator instructions regarding all processing applications. The guidance should explain how to perform particular jobs, including how operators should respond to system requests or interrupts. The documentation should only include information pertinent to the computer operator's function. Program documentation such as source listings, record layouts, and program flowcharts should not be accessible to an operator. Operator instructions should be thorough enough to permit an experienced operator who is unfamiliar with the application to run a program successfully without assistance.

- End-User Instructions – Organizations should establish end-user instructions that describe how to use an application. Operation manuals, online help features, and system error messages are forms of instructions that assist individuals in using applications and responding to problems.

TESTING PHASE

The testing phase requires organizations to complete various tests to ensure the accuracy of programmed code, the inclusion of expected functionality, and the interoperability of applications and other network components. Thorough testing is critical to ensuring systems meet organizational and end-user requirements.

If organizations use effective project management techniques, they will complete test plans while developing applications, prior to entering the testing phase. Weak project management techniques or demands to complete projects quickly may pressure organizations to develop test plans at the start of the testing phase. Test plans created during initial project phases enhance an organization's ability to create detailed tests. The use of detailed test plans significantly increases the likelihood that testers will identify weaknesses before products are implemented.

Testing groups are comprised of technicians and end users who are responsible for assembling and loading representative test data into a testing environment. The groups typically perform tests in stages, either from a top-down or bottom-up approach. A bottom-up approach tests smaller components first and progressively adds and tests additional components and systems. A top-down approach first tests major components and connections and progressively tests smaller components and connections. The progression and definitions of completed tests vary between organizations.

Bottom-up tests often begin with functional (requirements based) testing. Functional tests should ensure that expected functional, security, and internal control features are present and operating properly. Testers then complete integration and end-to-end testing to ensure application and system components interact properly. Users then conduct acceptance tests to ensure systems meet defined acceptance criteria.

Testers often identify program defects or weaknesses during the testing process. Procedures should be in place to ensure programmers correct defects quickly and document all corrections or modifications. Correcting problems quickly increases testing efficiencies by decreasing testers' downtime. It also ensures a programmer does not waste time trying to debug a portion of a program without defects that is not working because another programmer has not debugged a defective linked routine. Documenting corrections and modifications is necessary to maintain the integrity of the overall program documentation.

Organizations should review and complete user, operator, and maintenance manuals during the testing phase. Additionally, they should finalize conversion, implementation, and training plans.

Primary tests include:

- Acceptance Testing – End users perform acceptance tests to assess the overall functionality and interoperability of an application.
- End-to-End Testing – End users and system technicians perform end-to-end tests to assess the interoperability of an application and other system components such as databases, hardware, software, or communication devices.
- Functional Testing – End users perform functional tests to assess the operability of a program against predefined requirements. Functional tests include black box tests, which assess the operational functionality of a feature against predefined expectations, or white box tests, which assess the functionality of a feature's code.
- Integration Testing – End users and system technicians perform integration tests to assess the interfaces of integrated software components.
- Parallel Testing – End users perform parallel tests to compare the output of a new application against a similar, often the original, application.
- Regression Testing – End users retest applications to assess functionality after programmers make code changes to previously tested applications.
- Stress Testing – Technicians perform stress tests to assess the maximum limits of an application.
- String Testing – Programmers perform string tests to assess the functionality of related code modules.
- System Testing – Technicians perform system tests to assess the functionality of an entire system.
- Unit Testing – Programmers perform unit tests to assess the functionality of small modules of code.

IMPLEMENTATION PHASE

The implementation phase involves installing approved applications into production environments. Primary tasks include announcing the implementation schedule, training end users, and installing the product. Additionally, organizations should input and verify data, configure and test system and security parameters, and conduct post-implementation reviews. Management should circulate implementation schedules to all affected parties and should notify users of any implementation responsibilities.

After organizations install a product, pre-existing data is manually input or electronically transferred to a new system. Verifying the accuracy of the input data and security configurations is a critical part of the implementation process. Organizations often run a new system in parallel with an old system until they verify the accuracy and reliability of the new system. Employees should document any programming, procedural, or configuration changes made during the verification process.

Project Evaluation

Management should conduct post-implementation reviews at the end of a project to validate the completion of project objectives and assess project management activities. Management should interview all personnel actively involved in the operational use of a product and document and address any identified problems.

Management should analyze the effectiveness of project management activities by comparing, among other things, planned and actual costs, benefits, and development times. They should document the results and present them to senior management. Senior management should be informed of any operational or project management deficiencies.

MAINTENANCE PHASE

The maintenance phase involves making changes to hardware, software, and documentation to support its operational effectiveness. It includes making changes to improve a system's performance, correct problems, enhance security, or address user requirements. To ensure modifications do not disrupt operations or degrade a system's performance or security, organizations should establish appropriate change management standards and procedures.

Change management (sometimes referred to as configuration management) involves establishing baseline versions of products, services, and procedures and ensuring all changes are approved, documented, and disseminated. Change controls should address all aspects of an organization's technology environment including software programs, hardware and software configurations, operational standards and procedures, and project management activities. Management should establish change controls that address major, routine, and emergency software modifications and software patches.

Major modifications involve significant changes to a system's functionality. Management should implement major modifications using a well-structured process, such as an SDLC methodology.

Routine changes are not as complex as major modifications and can usually be implemented in the normal course of business. Routine change controls should include procedures for requesting, evaluating, approving, testing, installing, and documenting software modifications.

Emergency changes may address an issue that would normally be considered routine, however, because of security concerns or processing problems, the changes must be made quickly. Emergency change controls should include the same procedures as routine change controls. Management should establish abbreviated request, evaluation, and approval procedures to ensure they can implement changes quickly. Detailed evaluations and documentation of emergency changes should be completed as soon as possible after changes are implemented. Management should test routine and, whenever possible, emergency changes prior to implementation and quickly notify affected parties of all changes. If management is unable to thoroughly test emergency modifications before installation, it is critical that they appropriately backup files and programs and have established back-out procedures in place.

Software patches are similar in complexity to routine modifications. This document uses the term "patch" to describe program modifications involving externally developed software packages. However, organizations with in-house programming may also refer to routine software modifications as patches. Patch management programs should address procedures for evaluating, approving, testing, installing, and documenting software modifications. However, a critical part of the patch management process involves maintaining an awareness of external vulnerabilities and available patches.

Maintaining accurate, up-to-date hardware and software inventories is a critical part of all change management processes. Management should carefully document all modifications to ensure accurate system inventories. (If material software patches are identified but not implemented, management should document the reason why the patch was not installed.)

Management should coordinate all technology related changes through an oversight committee and assign an appropriate party responsibility for administering software patch management programs. Quality assurance, security, audit, regulatory compliance, network, and end-user personnel should be appropriately included in change management processes. Risk and security review should be done whenever a system modification is implemented to ensure controls remain in place.

Refer to the "Maintenance" section of this booklet and the *IT Handbook's* "Information Security Booklet" for additional details regarding change controls.

DISPOSAL PHASE

The disposal phase involves the orderly removal of surplus or obsolete hardware, software, or data. Primary tasks include the transfer, archiving, or destruction of data records. Management should transfer data from production systems in a planned and controlled manner that includes appropriate backup and testing procedures. Organizations should maintain archived data in accordance with applicable record retention requirements. It should also archive system documentation in case it becomes necessary to reinstall a system into production. Management should destroy data by

overwriting old information or degaussing (demagnetizing) disks and tapes. Refer to the *IT Handbook's* "Information Security Booklet" for more information on disposal of media.

LARGE-SCALE INTEGRATED SYSTEMS

Large-scale integrated systems are comprised of multiple applications that operate as an integrated unit. The systems are designed to use compatible programming languages, operating systems, and communication protocols to enhance interoperability and ease maintenance requirements.

Effectively implementing large-scale integrated systems is a complex task that requires considerable resources, strong project and risk management techniques, and a long-term commitment from corporate boards and management.

Although the anticipated benefits of integrated systems may be compelling, there are significant challenges associated with the development process. Some organizations underestimate the demands of such projects, incur significant financial losses, and ultimately abandon the projects.

Examiners encountering organizations that are implementing large-scale integrated systems must thoroughly review all life cycle procedures. The reviews should include an assessment of the board's understanding of project requirements and commitment to the project. Examiners must also closely review the qualifications of the project manager, the adequacy of project plans, and the sufficiency of risk management procedures.

SOFTWARE DEVELOPMENT TECHNIQUES

OBJECT-ORIENTED PROGRAMMING

Traditionally, programmers wrote programs using sequential lines of code in a high-level, procedural language such as COBOL⁶. Programmers also write object-oriented programs in high-level languages such as C++ and Java; however, the programs are written less sequentially.

Object-oriented programming centers on the development of small, reusable program routines (modules) that are linked together and to other objects to form a program. A key component of object-oriented programming involves the classification (modeling) of related data types (numbers, letters, dollars, etc.) and structures (records, files, tables, etc.). Modeling allows programmers to link reusable program modules to modeled data

⁶ High-level programming languages, such as COBOL (common business oriented language), are written using terminology similar to human languages. Java and C++ were developed to take advantage of object-oriented programming techniques. High-level programs are translated/compiled into lower-level languages so that hardware can read the code.

classes. Linking pre-developed program modules to defined data classes reduces development times and makes programs easier to modify.

Programmers use various methods to define and link reusable objects. Initially, structured programming techniques were developed that focused on the arrangement of lines of procedural code. The ad hoc techniques enhanced the layout of a program's modules and overall design, making it easier to integrate and reuse program modules. Object-oriented programming employs a form of structured programming and adds methods for defining the data classes that are linked to structured program routines.

A drawback to the use of structured programming, and consequently, methods such as object-oriented programming, which use structured programming techniques, has been a lack of standardized coding procedures. The lack of standardized procedures restricts the interoperability of proprietary products, including automated design and development products sometimes referred to as computer-aided software engineering (CASE) tools. However, the software industry is moving towards acceptance of standardized object-oriented modeling protocols.

COMPUTER-AIDED SOFTWARE ENGINEERING

CASE tools are a class of software that automates many of the activities involved in various life cycle phases. For example, when establishing the functional requirements of a proposed application, prototyping tools can be used to develop graphic models of application screens to assist end users to visualize how an application will look after development. Subsequently, system designers can use automated design tools to transform the prototyped functional requirements into detailed design documents. Programmers can then use automated code generators to convert the design documents into code. Automated tools can be used collectively, as mentioned, or individually. For example, prototyping tools could be used to define application requirements that get passed to design technicians who convert the requirements into detailed designs in a traditional manner using flowcharts and narrative documents, without the assistance of automated design software.

Automated tools can also facilitate the coordination of software development activities through the use of data warehouses or repositories. Repositories provide a means to store and access information relating to a project, such as project plans, functional requirements, design documents, program libraries, test banks, etc.

Organizations generally implement automated development tools to increase productivity, decrease costs, enhance project controls, and increase product quality. However, only by managing the various risks associated with automated technologies will organizations ensure they develop systems with appropriate functionality, security, integrity, and reliability.

Common CASE risks and associated controls include:

- **Inadequate Standardization** – Linking CASE tools from different vendors (design tool from Company X, programming tool from Company Y) may be difficult if the products do not use standardized code structures and data classifications. File formats can be converted, but usually not economically. Controls include using tools from the same vendor, or using tools based on standard protocols and insisting on demonstrated compatibility. Additionally, if organizations obtain tools for only a portion of the development process, they should consider acquiring them from a vendor that has a full line of products to ensure future compatibility if they add more tools.
- **Unrealistic Expectations** – Organizations often implement CASE technologies to reduce development costs. Implementing CASE strategies usually involves high start-up costs. Generally, management must be willing to accept a long-term payback period. Controls include requiring senior managers to define their purpose and strategies for implementing CASE technologies.
- **Quick Implementation** – Implementing CASE technologies can involve a significant change from traditional development environments. Typically, organizations should not use CASE tools the first time on critical projects or projects with short deadlines because of the lengthy training process. Additionally, organizations should consider using the tools on smaller, less complex projects and gradually implementing the tools to allow more training time.
- **Weak Repository Controls** – Failure to adequately control access to CASE repositories may result in security breaches or damage to the work documents, system designs, or code modules stored in the repository. Controls include protecting the repositories with appropriate access, version, and backup controls.

RAPID APPLICATION DEVELOPMENT

Rapid application development (RAD) is a software development technique that emphasizes short development times (30-90 days). The technique is inappropriate for developing complex applications or applications that quickly process significant volumes of transactions, such as batch processing environments. Organizations may appropriately use the technique to develop or redesign lower-risk applications, or less complex applications such as transactional websites that do not involve a high degree of throughput. Additionally, depending on an organization's risk tolerance and the mission-critical designation of an application, organizations may use RAD techniques during the design and development phases within a structured development methodology.

Management should consider the functional complexity and security risks of an application when selecting a development methodology. Management should ensure that the development technique selected is appropriate for managing the complexity and risks of the application being developed.

The RAD process may involve only three phases: initiation, development, and implementation. The short duration of RAD projects necessitates the quick identification of functional requirements, which should remain largely unchanged during the development process. Typically, managers assign end users full time to a project and empower them to make key design decisions. However, they normally consult project specialists such as database administrators, network technicians, and system programmers, when they make key decisions.

End users and RAD team members usually create functional designs using prototyping tools in an iterative process. They create, review, and revise the prototypes as needed until they approve a final design.

RAD methodologies often employ object-oriented programming techniques that take advantage of reusable program objects. In the case of redesigned applications, designers and developers select objects based on the function they perform and re-link them to other reusable or newly created objects to form a new application.

Testing often occurs concurrently with the development of functional features. Organizations may conduct testing and implementation procedures quickly or in a structured process similar to SDLC testing and implementation phases. The speed and structure of testing and implementation procedures depends on an organization's risk tolerance, an application's mission-critical designation, and a project's established deadlines.

Developers are increasingly creating web applications using RAD techniques. Numerous CASE-type products (for example, application program interfaces) are available that developers can use to combine pre-designed object-oriented functions to form unified applications.

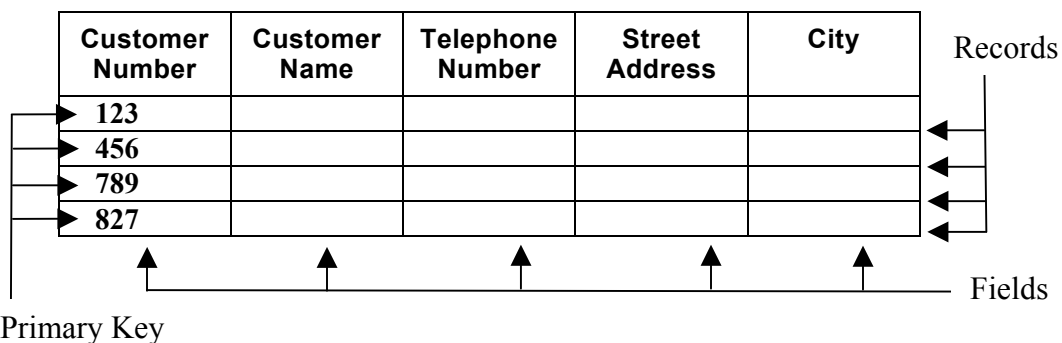
Appropriate standards and controls should be in place to ensure:

- Organizations only use RAD techniques when appropriate;
- Management includes adequate security and control features in all developed applications;
- Quality assurance personnel verify that security and control features (commensurate with risk levels) exist and function as intended;
- End users are appropriately involved throughout RAD projects; and
- Project managers closely monitor project activities.

DATABASES

Databases contain stored information and are structured in various ways. Legacy systems often use hierarchical or networked structures. Hierarchical databases are structured similar to an organizational chart. Lower data elements are dependent on (communicate data between) a single data element above them. Each data element can have multiple elements linked below it, but only one link to a data element above it. Networked databases are similar to hierarchical databases, but data elements can link to multiple elements above or below them.

Relational databases, which are currently the most prevalent type database, are organized as tables with data structured in rows and columns. (Data paths are not predefined because relationships are defined at the data value level.) Rows (or records) contain information that relates to a single subject, such as a customer, employee, or vendor. Columns (or fields) contain information related to each subject, such as customer identification numbers, dates of birth, or business addresses. Each record (item in a row), links to a corresponding field element (item in a column) that is defined as the primary key. Primary keys are the main identifiers associated with stored information and facilitate access to the information. In certain situations, the primary key may be comprised of data from more than one field.



Relational databases are usually comprised of multiple tables, which may reside on a single server or in a distributed environment. If related records are stored on multiple tables (for example, if a customer's primary information, as shown in the table above, is maintained at a customer service center and the customer's deposit account information is maintained on a second table/database at a local branch), the same primary key must be used in both tables to ensure data integrity. The keys in secondary tables, however, are referred to as foreign keys.

Object-relational databases have been developed that apply ad hoc object-oriented protocols in relational database environments. Definitive standards do not currently exist to support the wide acceptance of strictly object-oriented databases and database management systems. However, various proprietary standards do exist, and organizations are attempting to develop standardized object-oriented database protocols.

DATABASE MANAGEMENT SYSTEMS

Database management systems (DBMS) are software programs that control a database user's access and modification rights. The systems also facilitate referential integrity (by managing cross references between primary and foreign key relationships), support data import and export functions, and provide backup and recovery services.

Database management systems may also provide access to data dictionaries. Data dictionaries are documentation tools that store descriptions of the structure and format of data and data tables. Advanced data dictionaries may store source code copies of field, record, and code descriptions for use during software design and development activities.

Primary issues to consider when reviewing the design and configuration of database management systems include access controls and auditing features. Management should restrict direct (privileged) access to a database (as opposed to accessing information through an application) to authorized personnel.

Most DBMS have a journaling feature that allows organizations to track data changes. Journaling provides audit trails of data changes and facilitates the safe recovery of data if errors occur. If available, organizations should employ automated auditing tools, such as journaling, that identify who accessed or attempted to access a database and what, if any, data was changed.

Many DBMS can validate users at record and row levels and log their activities. The detailed validation levels provide strong security controls. Examiners should consider validation levels when assessing the adequacy of DBMS controls. Strong DBMS controls include data-change logs, input validity checks, locking and rollback mechanisms (ability to recover a previous database if the database becomes corrupted), password and data file encryption. System developers should consider incorporating these types of security features when designing databases. If strong controls or auditing features are unavailable, management should implement compensating controls such as segregation-of-duty or dual controls.

ACQUISITION

Action Summary

Financial institutions should establish appropriate acquisition methodologies. The methodologies should match a project's characteristics and risks and require appropriate:

- Project plans;
- Project standards and procedures;
- Quality assurance, risk management, and testing standards and procedures;
- Definitions of product requirements;
- Involvement by all affected parties;
- Vendor, contract, and license reviews; and
- Escrow documentation.

Acquisition projects are similar to development projects because management approves project requests, defines functional, security, and system requirements, and appropriately tests and implements products. Organizations often employ structured acquisition methodologies similar to the SDLC when acquiring significant hardware and software products. However, organizations replace the SDLC design and development phases with a bid solicitation process that involves developing detailed lists of functional, security, and system requirements and distributing them to third parties. The "Acquisition Project Guidance" discussion below centers on the specific activities associated with acquisition projects. Refer to the Project Management and Development sections for additional details relating to general life cycle phase information.

In addition to developing and distributing detailed lists of functional, security, and system requirements, organizations should establish vendor selection criteria and review potential vendors' financial strength, support levels, security controls, etc., prior to obtaining products or services. Additionally, management reviews contracts and licensing agreements to ensure the rights and responsibilities of each party are clear and equitable. Primary risks include inadequately defining requirements, ineffectively assessing vendors, and insufficiently reviewing contracts and agreements.

Contract and licensing issues may arise due to the complexity of contractual requirements. An organization's legal counsel should confirm that performance guarantees, source code accessibility, intellectual property considerations, and software/data security issues are appropriately addressed before management signs contracts.

Financial institutions sometimes acquire software or services from foreign-based third parties. Organizations should appropriately manage the unique risks included in these arrangements. For example, organizations should decide which country's laws will control the relationship and ensure they and their vendors comply with United States' laws that restrict the export of software applications employing encryption techniques. Refer to the "Software Development Contracts and Licensing Agreements" discussion for additional details on contracts and licenses. Refer to the *IT Handbook's* "Outsourcing Technology Services Booklet" for additional information relating to foreign-based third-party relationships.

ACQUISITION STANDARDS

Management should establish acquisition standards that address the same security and reliability issues as development standards. However, acquisition standards should focus on ensuring security, reliability, and functionality are already built into a product. Acquisition standards should also ensure managers complete appropriate vendor, contract, and licensing reviews and acquire products compatible with existing systems.

Key tools in managing acquisition projects include invitations-to-tender and request-for-proposals. Invitations-to-tender involve soliciting bids from vendors when acquiring hardware or integrated systems of hardware and software. Request-for-proposals involve soliciting bids when acquiring off-the-shelf or third-party developed software. However, the terms are sometimes used interchangeably.

Management should establish acquisition standards to ensure functional, security, and operational requirements are accurately identified and clearly detailed in request-for-proposals and invitations-to-tender. The standards should also require managers to compare bids against a project's defined requirements and against each other; to review potential vendors' financial stability and commitment to service; and to obtain legal counsel reviews of contracts before management signs them.

Note: The risks associated with using general business purpose, off-the-shelf software, such as a word processing application, are typically lower than those associated with using financial applications. Therefore, the acquisition of general business purpose, off-the-shelf software typically requires less stringent evaluation procedures than acquiring hardware or software specifically designed for financial purposes. However, the level of evaluation will depend on how risky the application is and how critical it is to the institution.

ACQUISITION PROJECT GUIDANCE

Acquisition projects begin with the submission of a project request. Procedures should be in place to facilitate the request process and ensure management systematically reviews all requests. Requests should present a business case for acquiring a product, identify desired system features, and, to the extent possible, describe the information requirements, network interfaces, hardware components, and software applications that will support and interact with a new product. Management should complete a feasibility study to determine if the business case supports the procurement of either customized or off-the-shelf software. All affected parties should document their approval of the overall feasibility of the project.

Determining the feasibility of an acquisition proposal includes consideration of issues such as:

- Business objectives;
- Technology objectives;
- Functional requirements;
- Security requirements;
- Internal control requirements;
- Documentation requirements;
- Performance requirements;
- Network requirements;
- System interface requirements;
- Expandability requirements;
- Reliability requirements;
- Maintenance requirements;
- Installation requirements;
- Conversion requirements;
- Personnel requirements;
- Processing requirements;
- Product development standards;
- Product design standards;
- Testing requirements;
- Training requirements;
- Vendor's financial strength;
- Vendor's support levels; and
- Cost/benefit analysis.

To determine the feasibility of a project, management should consult with various personnel, including those listed below. These individuals should be involved in all phases of the project as deemed appropriate depending on their role in relation to the specific system being acquired:

- Audit personnel;
- Business unit managers;
- Database administrators;
- End users;
- Legal counsel;
- Network administrators;
- Network technicians;
- Quality assurance personnel;
- Security administrators;
- Systems analysts;
- Technology department managers; and
- Vendor personnel.

If a request appears feasible, the feasibility study can help define the functional, system, and organizational requirements included in the request-for-proposals and invitations-to-tender that management distributes to third parties in the bid solicitation process.

After organizations receive bids they should analyze and compare the bids against each other and against the organization's defined requirements. Vendors' proposals should clearly address all of the organization's requirements and identify any other applicable issues such as:

Software:

- Confidentiality standards;
- Compatible operating systems;
- Copyright standards;
- Delivery dates;
- Escrow criteria;
- Liability limitations;
- Licensing restrictions;
- Maintenance procedures;
- Next release date;
- Regulatory requirements;
- Software language;
- Subcontractor details;

- Testing standards;
- Training provided; and
- Warranty specifications.

Hardware:

- Backup options;
- Maintenance requirements;
- Memory capacities;
- Performance capabilities; and
- Servicing options.

Procedures should be in place to ensure organizations appropriately review bids. After the selection process narrows the list of potential vendors, management should review the financial stability and service commitment of the remaining vendors. After an organization selects a product and vendor and negotiates a contract, legal counsel should review the contract prior to signing.

ESCROWED DOCUMENTATION

Software programs are written using non-proprietary, open source code; proprietary (licensed) open source code; or proprietary, closed source code.

Non-proprietary, open source programs, sometimes referred to as free software, are written in publicly available code and can usually be used, copied, modified, etc., without restriction. Proprietary, open source programs are also written in publicly available code but are copyrighted and distributed through various licensing agreements. Management should carefully consider all licensing agreements to ensure their use, modification, or redistribution of the programs conforms to the agreements.⁷

Proprietary, closed source programs are normally copyrighted trade secrets of the company that wrote or owns the programs. Most vendors do not release closed source code to the organizations that buy or lease the products in order to protect the integrity and copyrights of the software. An alternative to receiving the source information is to install programs in object code and establish a source code escrow agreement. In such an agreement, organizations can only access the source code under specific conditions, such as discontinued product support or financial insolvency of the vendor.

⁷ For additional information, organizations that use, or are considering using, open source software should consult with their legal staff and review open source definitions, licensing standards, certification criteria, and related information distributed by various organizations (for example, the Open Source Initiative at www.opensource.org). (Note: Reference to this organization is for illustrative purposes only and is not an endorsement of the organization.)

Typically, an independent third party retains the documentation in escrow; however it is each organization's responsibility to periodically (at least annually) ensure the third party holds a current version of the source information. Often, escrow agents provide services for reviewing and confirming source code version numbers and dates. Some agents also perform automated code reviews to ensure the integrity of the escrowed code.⁸

In addition to ensuring access to current documentation, organizations should consider protecting their escrow rights by contractually requiring software vendors to inform the organization if the software vendor pledges the software as loan collateral.

Provisions management should consider incorporating into escrow agreements include:

- Definitions of minimum programming and system documentation⁹;
- Definitions of software maintenance procedures;
- Conditions that must be present before an organization can access the source information;
- Assurances that the escrow agent will hold current, up-to-date versions of the source programs and documentation (escrowed information must be updated whenever program changes are made);
- Arrangements for auditing/testing the escrowed code;
- Descriptions of the source information media (for example, magnetic tape, disc, or hard copy) and assurances that the media is operable and compatible with an organization's existing technology systems;
- Assurances that source programs will generate executable code.

SOFTWARE DEVELOPMENT CONTRACTS AND LICENSING AGREEMENTS

OVERVIEW

Contracts between an organization and a software vendor should clearly describe the rights and responsibilities of the parties to the contract. The contracts should be in writing with sufficient detail to provide assurances for performance, source code accessibility, software and data security, and other important issues. Before management signs the contracts, it should submit them for legal counsel review.

⁸ Management should consider the practical and legal implications of establishing escrow arrangements with foreign-based entities when determining the feasibility of foreign-based relationships.

⁹ At minimum, the software documentation held by the escrow agent should include system narratives, system flow charts, program source listings, program narratives, file and record layouts, descriptions of individual fields within the records, and calculation routines. Portions of this documentation may be included with the user guides that are provided to the financial institution. These documents should also cover transaction codes and descriptions of input forms and output reports.

Organizations may encounter situations where software vendors cannot or will not agree to the terms an organization requests. Under these circumstances, organizations should determine if they are willing to accept or able to mitigate the risks of acquiring the software without the requested terms. If not, consideration of alternative vendors and software may be appropriate.

SOFTWARE LICENSES - GENERAL

Software is usually licensed, not purchased. Under licensing agreements, organizations obtain no ownership rights, even if the organization paid to have the software developed. Rather, the software developer licenses an organization certain rights to use the software. Vendors typically grant license rights for a specific time period and may require annual fees to use the software. Vendors may also provide maintenance agreements that assure they will provide new versions or releases of the software.

The most important licensing issue is the definition of the precise scope of the license. Organizations should ensure that licenses clearly state whether software usage is exclusive or not exclusive, who or how many individuals at the organization can use the software, and whether there are any location limitations on its use. Before negotiating a license, organizations should accurately assess current and future software needs and ensure the license will meet their needs.

The license should clearly define permitted users and sites. If an organization desires a site license for an unlimited number of users at its facilities, it should ensure the contract expressly provides for this. If the organization requires other related entities to use the software, such as subsidiaries or contractors, they should also be included in the license. Organizations should also ensure they have an express license to retain and use backup copies of any mission-critical software that they may need to carry out disaster recovery or business continuity programs at remote sites.

Organizations should clearly understand the duration of the licenses to prevent unexpected license expirations. If an organization desires a perpetual license to use the software, it should ensure the contract explicitly grants such a license. Organizations should not assume the failure to specify a fixed term or termination date automatically provides them with a perpetual license. At a minimum, organizations should specify in their contract or license the time periods for a non-perpetual license and the minimum amount of notice required for termination.

SOFTWARE LICENSES AND COPYRIGHT VIOLATIONS

Copyright laws protect proprietary as well as open-source software. The use of unlicensed software or violations of a licensing agreement expose organizations to possible litigation.

Management should take particular caution when purchasing software for use on a network. Some programs are not licensed for shared use and management may be required to purchase individual copies for each network user. Additionally, some network licenses only allow a predetermined number of persons to use the programs concurrently.

Measures that organizations may employ to protect against copyright violations include obtaining a site license that authorizes software use at all organization locations, informing employees of the rules governing site licenses, and acquiring a software management program that scans for unauthorized software use or copyright violations. While these measures may help prevent copyright violations, the best control mechanism is a strict corporate policy that management and auditors communicate and enforce. Management should have an uncompromising attitude regarding copyright violations. The organization's security administrator should be responsible for monitoring and enforcing the policy.

SOFTWARE DEVELOPMENT SPECIFICATIONS AND PERFORMANCE STANDARDS

Contracts for the development of custom software should describe and define the expected performance attributes and functionality of the software. The contract should also describe the equipment required to operate the software to ensure appropriate compatibility. Vendors should be required to meet or exceed an institution's internal development policies and standards. Therefore, before opening negotiations or issuing a request-for-proposal on custom software development, organizations should have a clear idea of the essential business needs to be addressed by the software and an adequate understanding of the organization's present and planned system architectures.

Contracts should identify and describe the functional specifications that operational software will perform and may identify functional milestones that vendors must meet during the development process. The development contract should also contain provisions that permit the modification of specifications and performance standards during the development process.

Software development contracts should contain objective pre-acceptance performance standards to measure the software's functionality. The contracts may identify particular tests needed to determine whether the software complies with performance standards. The contracts may also address what actions a vendor will take if the software fails one or more tests.

DOCUMENTATION, MODIFICATION, UPDATES, AND CONVERSION

A licensing or development agreement should require vendors to deliver appropriate software documentation. This should include both application and user documentation.

A license or separate maintenance agreement should address the availability and cost of software updates and modifications. When drafting agreements, organizations should determine if a vendor provides access to source or object code. Regardless of whether vendors limit access to object code or provide access to source code, the permission and willing participation of the vendor may be necessary to make modifications to the software. Modifications to source code may void maintenance agreements.

When negotiating a software license, organizations should anticipate they might need to convert to a different software product in the future. The license should enable and facilitate conversions. The license should not restrict an organization's ability to convert data to a new format. If possible, organizations should negotiate terms that would enable another firm to access the software and assist them in the conversion without violating license restrictions.

BANKRUPTCY

In addition to escrow agreements, organizations should consider the need for other clauses in licensing agreements to protect against the risk of a vendor bankruptcy. For mission-critical software, organizations should consult with their legal counsel on how best to deal with Section 365(n) of the Bankruptcy Code, which gives a bankrupt vendor discretion to determine which of its executory contracts it will continue to perform and which it will reject. Proper structuring of the agreement can help an organization protect its interests if a vendor becomes insolvent.

REGULATORY REQUIREMENTS

Depending on the function of the specific software, organizations should consider including a regulatory requirements clause in their licensing agreements. The clause requires vendors to maintain application software so that functions are performed in compliance with applicable federal and state regulations.

PAYMENTS

Software development contracts normally call for partial payments at specified milestones, with final payment due after completion of acceptance tests. Organizations should structure payment schedules so developers have incentives to complete the project quickly and properly. Properly defined milestones can break development projects into deliverable segments so an organization can monitor the developer's progress and identify potential problems.

Organizations should exercise caution when entering into software development contracts that base compensation on the developer's time and materials. A fixed price agreement with specific payment milestones is sometimes preferable because it provides an organization more control over the development process and total project costs.

Contracts should detail all features and functions the delivered software will provide. If a vendor fails to meet any of its express requirements, organizations should retain the right to reject the tendered product and to withhold payment until the vendor meets all requirements.

REPRESENTATIONS AND WARRANTIES

Organizations should seek an express representation and warranty in the software license that the licensed software does not infringe upon the intellectual property rights of any third parties worldwide. Under some state laws, non-infringement warranties are limited to the United States unless otherwise specifically provided.

Vendors should also represent and warrant that software will not contain undisclosed restrictive code or automatic restraints not specifically authorized in the agreement. (See discussion under "Security".)

Licenses should also include appropriate warranties that software will perform according to specifications and should state how a vendor will respond in the event of problems. Warranties should distinguish between mission-critical failures, which require an expedited response, and failures that are not critical, which an organization can resolve in a routine manner. Licenses should also specify the length of the warranty and how the warranty relates to maintenance obligations and agreements.

DISPUTE RESOLUTION

Organizations should consider including dispute resolution provisions in contracts and licensing agreements. Such provisions enhance an organization's ability to resolve problems expeditiously and may provide for continued software development during a dispute resolution period.

AGREEMENT MODIFICATIONS

Organizations should ensure software licenses clearly state that vendors cannot modify agreements without written signatures from both parties. This clause helps ensure there are no inadvertent modifications through less formal mechanisms some states may permit.

VENDOR LIABILITY LIMITATIONS

Some vendors may propose contracts that contain clauses limiting their liability. They may attempt to add provisions that disclaim all express or implied warranties or that limit monetary damages to the value of the product itself, consideration paid, or specific liquidated damages. Generally, courts uphold these contractual limitations on liability in commercial settings unless they are unconscionable. Therefore, if organizations are considering contracts that contain such clauses, they should consider whether the proposed damage limitation bears an adequate relationship to the amount of loss the financial organization might reasonably experience as a result of the vendor's failure to perform its obligations. For mission-critical software, broad exculpatory clauses that limit a vendor's liability are a dangerous practice that could adversely affect the soundness of an organization because organizations could be substantially injured and have no recourse.

SECURITY

Organizations should develop security control requirements for information systems and incorporate performance standards relating to security features in their software licensing and development contracts. The standards should ensure software is consistent with an organization's overall security program. In developing security standards, organizations may wish to reference the methodology detailed in the *IT Handbook's* "Information Security Booklet". Organizations may also refer to other widely recognized industry standards.

Contracts should also address a vendor's ongoing responsibilities to protect the security and confidentiality of an organization's resources and data. The agreement should prohibit vendors and their contractors and agents from using or disclosing an organization's information except as necessary to provide contracted services. Further, organizations should seek a guaranty from vendors that software does not and will not contain any back doors or disabling devices that would permit unauthorized access to the application or any of the organization's systems or data. For mission-critical software, contracts and licenses should explicitly state that the vendor will not use software features that enable them to remotely disable software in the event of a dispute with the purchaser. Additionally, contracts and licenses should state that the organization may only be deprived of its software use through a court order.

Software development packages may include significant update, modification, training, operational, and support services that require a vendor's access to an organization's customer data. These aspects of the relationship trigger service provider requirements under the federal banking agencies' "Interagency Guidelines Establishing Standards for Safeguarding Customer Information" that implement Section 501(b) of the Gramm-Leach-Bliley Act. The guidelines expressly state that organizations shall require service

providers by contract to implement appropriate measures designed to meet the security objectives of the guidelines.

SUBCONTRACTING AND MULTIPLE VENDOR RELATIONSHIPS

Some software vendors may contract third parties to develop software for their clients. To provide accountability, it may be beneficial for organizations to designate a primary contracting vendor. Organizations should include a provision specifying that the primary contracting vendor is responsible for the software regardless of which entity designed or developed the software. Organizations should also consider imposing notification and approval requirements regarding changes in vendor's significant subcontractors. Refer to the *IT Handbook's* "Outsourcing Technology Services Booklet" for additional subcontracting information.

Organizations should consider contract provisions that prohibit the assignment of contracts by vendors to a third party without the organization's consent. Conversely, organizations that expect to be acquired or restructured should determine whether their licensing agreements continue after the transition. Some software license agreements contain change-of-control or transfer limitations that inhibit use of the software after a merger, acquisition, or change of ownership.

RESTRICTIONS ON ADVERSE COMMENTS

Some software licenses include a provision prohibiting licensees from disclosing adverse information about the performance of the software to any third party. Such provisions could inhibit an organization's participation in user groups, which provide useful shared experience regarding software packages. Accordingly, organizations should resist these types of provisions.

MAINTENANCE

Action Summary

Financial institutions should establish appropriate maintenance methodologies. The methodologies should match a project's characteristics and risks and require appropriate:

- Project planning;
- Maintenance standards and procedures;
- Major, routine, and emergency change controls;
- Patch management controls;
- Involvement by all affected parties;
- Documentation standards;
- Library and utility controls; and
- Quality assurance and risk management standards and procedures.

Maintenance activities include the routine servicing and periodic modification of hardware, software, and related documentation. Hardware modifications are periodically required to replace outdated or malfunctioning equipment or to enhance performance or storage capacities. Software modifications are required to address user requirements, rectify software problems, correct security vulnerabilities, or implement new technologies. Documentation maintenance is necessary to maintain current, accurate, technology-related records, standards, and procedures.

Failure to implement appropriate change controls can result in operational disruptions or degrade a system's performance or security. Change controls (sometimes referred to as configuration management) involve establishing baseline versions of products, services, or procedures and ensuring all changes are approved, documented, and disseminated. Change controls should address all aspects of an organization's technology environment including software programs, hardware and software configurations, operational standards and procedures, and project management activities.

Change controls can be applied universally to all systems and environments or stratified to particular systems, business lines, support areas, etc. Stratified procedures are often necessary to address the distinct control requirements of mainframe, network, and client/server environments, operating and application programs, and development and acquisition projects.

Management should establish detailed change control standards and procedures to ensure technology related modifications are appropriately authorized, tested, documented,

implemented and disseminated. The characteristics and risks of a system, activity, or change should dictate the formality of the change controls. Quality assurance, security, audit, network, and end-user personnel should be appropriately involved in the change process.

MAJOR MODIFICATIONS

Major modifications include significant functional changes to an existing system, converting to a new system, and introducing new systems or data due to corporate mergers or acquisitions. Major modifications should be implemented following a well-structured process similar to the SDLC (systems development life cycle) described in the "Development" section.

Major modification standards should address applicable project activities such as requirements analysis, feasibility studies, project planning, software design, programming, testing, and implementation procedures. Refer to the "Project Management" and "Development" sections for additional project management details. Refer to the "Conversion" discussion later in this section for specific considerations relating to major modifications.

ROUTINE MODIFICATIONS

Routine modifications involve making changes to application or operating system software to improve performance, correct problems, or enhance security. Routine modifications can be simple or complex, but are not of the magnitude of major modifications and can be implemented in the normal course of business.

Routine change standards should include change request, review, and approval procedures and require management to plan, test, and document all changes prior to implementation. Well defined implementation plans, which often include automated deployment tools, are especially important for large organizations that must implement changes over numerous or widely dispersed networks. Change standards should also address communication procedures to ensure management quickly notifies affected parties of all changes.

Organizations should coordinate software modifications and patches through a centralized change management process. Centralized oversight is necessary due to the interdependence of technology systems and operations. Large institutions should consider using specialized change control committees to coordinate activities. Smaller institutions can often use technology steering committees to effectively manage the process. Oversight committees help clarify request requirements and help ensure all departments are aware of pending changes. The committees should include sufficient representation from business, technology, security, quality assurance, and audit departments to ensure changes support business objectives and do not adversely affect operations or security.

Management should review all proposed changes to ensure modifications are appropriate for the system involved. Additionally, management should ensure modified programs are compared to change authorization documents to determine that only approved changes were implemented. The absence of sound controls and accurate documentation can cause problems when management installs subsequent systems. Standard change request forms, library and version controls, and spreadsheets or automated change logs facilitate management's ability to track, report, and analyze changes. Comprehensive change logs are a prerequisite to all change control processes.

Change request forms should provide an accurate chronological record and description of all changes. The forms should provide sufficient information for affected parties to understand the impact of a change and include:

- Request date;
- Requestor's name;
- Description of change;
- Reasons for implementing or rejecting a change;
- Justification for change;
- Approval signature(s); and
- Change control number.

If a change request is approved, the request form should be submitted to the appropriate technology department. The organization should develop additional documentation during the change process that includes:

- Priority information;
- Identification of affected systems, databases, and departments;
- Name of individual responsible for making the change;
- Resource requirements;
- Projected costs;
- Projected completion date;
- Projected implementation date;
- Potential security and reliability considerations;
- Testing requirements;
- Implementation procedures;
- Estimated downtime for implementation;
- Backup/Back-out procedures;
- Documentation updates (program designs and scripts, network topologies, user manuals, contingency plans, etc.);
- Change acceptance documentation from all applicable departments (user, technology, quality assurance, security, audit, etc.); and

- Post-implementation audit documentation (comparison of expectations and results).

After program modifications are completed, all program codes (source code, object code, patch code, load module, etc.) should be secured. Securing the codes provides some assurance that the programs cataloged to production environments are unaltered versions of the approved and tested programs. Management should establish program approval standards that include procedures for verifying test results, inspecting modified code, and confirming source and object codes match.

EMERGENCY MODIFICATIONS

Emergency modifications are periodically needed to correct software problems or restore processing operations quickly. Although the changes must be completed quickly, they should also be implemented in a well-controlled manner.

Emergency change standards should include procedures similar to those for routine change controls. However, the standards should include abbreviated change request, evaluation, and approval procedures to ensure changes are made quickly. The standards should be designed to ensure management completes detailed evaluations and documentation of emergency changes as soon as possible after implementation.

Whenever possible, emergency changes should be tested prior to implementation. If management is unable to thoroughly test emergency modifications before installation, it is critical that they appropriately backup files and programs and have established back-out procedures in place.

Appropriate backups, established back-out procedures, and detailed documentation enhance management's ability to reverse changes if they cause system disruptions. Detailed documentation also enhances management's ability to analyze the impact of any changes during post-change evaluations. At a minimum, emergency change procedures should require:

- Pre-change reviews and authorizations;
- Pre-change testing (in segregated testing environments);
- Backup/backout procedures;
- Documentation that includes:
 - Descriptions of a change;
 - Reasons for implementing or rejecting a proposed change;
 - The name of the individual who made the change;
 - A copy of the changed code;
 - The date and time a change was made; and
- Post-change evaluations.

PATCH MANAGEMENT

Software patches are defined in this document as program modifications involving externally developed software. Patch management standards should include procedures (similar to the routine modification standards described above) for identifying, evaluating, approving, testing, installing, and documenting patches.

Vendors frequently develop and issue patches to correct software problems, improve performance, and enhance security. Organizations should have procedures in place to identify available patches and to acquire them from trusted sources. Procedures for identifying software vulnerabilities and patch information include subscribing to patch-alert e-mail lists and monitoring vendor and security related websites. Management should regularly obtain bulletins about product enhancements and security issues as well as available patches and upgrades from its vendors or other trusted information security sources.

When an available patch is identified, management should evaluate the impact of installing the patch by assessing technical, business, and security implications. If management identifies a significant patch but decides not to install it, they should document their reasons for not installing it.

In order to minimize operational disruptions, management should test all patches prior to implementation. Additionally, management should appropriately backup files and programs and have established back-out procedures in place before implementation.

As with all software modifications, appropriate backup and back-out procedures, post-implementation evaluations, detailed documentation, and established implementation plans enhance management's ability to effectively control patch activities.

Note: The installation of software patches may reset security settings or configuration parameters to default settings. Management should review all settings and parameters after patches are applied to ensure the settings conform to approved policies and procedures.

LIBRARY CONTROLS

Libraries are collections of information, typically segregated by the type of stored information, such as development, testing, and production-related programs, data, or documentation.

Management should strictly control access to all libraries and the movement of programs and files between libraries. Programming personnel should not move programs into or out of production libraries. Library controls provide ways to manage the movement of programs between development, testing, and production environments. Management should assign librarian functions to independent quality assurance and production control personnel in larger institutions or to supervisory personnel in smaller institutions.

Commensurate with the complexity of their technology environments, organizations should consider using automated change controls. Regardless of the use of automated change control tools, management should strictly control access to production software libraries, particularly in distributed environments.

Management should establish appropriate controls to manage the movement of modified programs between libraries. The controls should include:

- Assignment of library custodian responsibilities;
- Verification of program integrity before programs are transferred to production libraries;
- Approval procedures for promoting programs into production;
- Password controls on all libraries or objects within libraries; and
- Automated library programs that restrict library access and identify who accessed a library and what, if any, changes were made.

CONVERSIONS

Conversions include major changes to existing applications or systems and the introduction of new systems or data resulting from corporate mergers or acquisitions. Conversions involve complex system changes that typically span multiple platforms. The elevated complexity increases risk levels that require detailed, systematic controls. Strong conversion controls are critical for preventing data corruption, performance degradation, and operational disruptions. Poorly controlled conversions can result in security or accounting problems, user or customer dissatisfaction, or reputation damage.

Conversions impact operational activities; therefore, management should closely evaluate all technology operations and determine if a proposed conversion is feasible and supports organizational objectives. Successful conversions require management to use a number of control disciplines including strategic planning, project management, requirements definition, testing, implementation, contingency planning, vendor management, and post-implementation reviews.

Conversion controls should include documented project plans, structured project management techniques, and comprehensive senior management oversight. Institutions often implement major conversions using a project management methodology similar to the SDLC process described in the Development section. All institutions engaged in frequent acquisitions or mergers should use standardized conversion methodologies implemented by specialized conversion teams.

Effective conversion management begins with due diligence that includes a comprehensive analysis of a conversion's impact on existing operations. Management should assess current and projected transaction, data storage, communication, and processing requirements. Managers should carefully assess increased demands for balancing, reconciliation, exception handling, problem resolution, user and customer

support, network connectivity, and system administration to ensure they are able to effectively complete conversions.

Organizations should also consider training requirements associated with conversions or major hardware/software upgrades. Management should evaluate the type, volume, and timing of training needs for each affected business unit and coordinate training programs with applicable vendors.

Successful conversions require close cooperation within an organization and between an organization and its vendors. Management should establish communication procedures, reporting requirements, and lines of authority to ensure they can make and communicate decisions quickly.

Successful conversions also require detailed file mapping. Technical, operational, and business unit personnel from all organizations involved in a merger or acquisition may be required to assist in the mapping process. Organizations should have a comprehensive knowledge of products at both institutions so they can map and transfer data from the converted institution to the surviving system. Failure to map files and accounts appropriately can result in customer or employee frustration, compliance issues, reputation damage, and possibly the loss of customers. All conversions should include adequate testing.

UTILITY CONTROLS

Standards should be in place to control the use of utility programs. Utility programs facilitate the management of disk drives, printers, and other operating system resources. Utility programs often exist as part of an operating system, and provide tools for program debugging, file maintenance, and corrupt data correction. Utility programs can allow a user to create, modify, move, rename, copy, or delete code and data from program libraries and data files. Therefore, management should strictly control the use of the programs.

Organizations should restrict the use of utility programs with logical access controls or remove the utilities from program libraries, place them under dual control, and load them only when necessary. Management should exercise caution to ensure the removal of a utility does not adversely affect other systems or applications.

DOCUMENTATION MAINTENANCE

Documentation provides valuable descriptions of an organization's development, acquisition, and operating environments and significantly enhances an organization's ability to administer, operate, and maintain technology systems. Primary advantages for end users involve having access to operation manuals and on-line application help features. Documentation enhances administrators' and technicians' ability to maintain and update systems efficiently and to identify and correct programming defects.

Developing and maintaining current, accurate documentation can be complicated, time consuming, and expensive. However, standardized documentation procedures and the use of automated documentation software facilitate an organization's ability to maintain accurate documentation.

Documentation standards should identify primary documentation custodians and detail document authoring, approving, and formatting requirements. Personnel should document all changes to system, application, and configuration documentation according to prescribed standards. Additionally, management should control access to documentation libraries with appropriate library and version controls.

APPENDIX A: EXAMINATION PROCEDURES

EXAMINATION OBJECTIVES: The objectives of Development and Acquisition assessments are to identify weaknesses or risks that could negatively impact an organization, to identify entities whose condition or performance requires special supervisory attention, and to subsequently effect corrective action.

Examiners should not expect organizations to employ formal project management techniques in all situations. Reviews should be risk focused and center on ensuring project management standards, controls, and procedures are present and commensurate with the characteristics and risks of the projects under review.

Examiners are not required to include lengthy responses to each Objective or bulleted item. Often, examiners should be able to simply note an item is adequate or inadequate with yes or no responses. However, examiners must adequately document material findings. Documentation must be sufficient to support the assignment of the Development and Acquisition component rating of the Federal Financial Institutions Examination Council's Uniform Rating System for Information Technology.

OBJECTIVES AND PROCEDURES

Objective 1: Determine the scope of the Development and Acquisition review.

1. Identify strengths and weaknesses relating to development, acquisition, and maintenance activities, through a review of:
 - Prior reports of examination;
 - Internal and external audits;
 - Regulatory, audit, and security reports from key service providers;
 - Organizational charts;
 - Network topology maps; and
 - Résumés of technology managers.
2. Review management's response to report and audit findings to determine:
 - The adequacy and timing of corrective actions;
 - The resolution of root causes rather than just specific issues; and
 - The existence of outstanding issues.
3. Review applicable documentation and interview technology managers to identify:
 - The type and frequency of development, acquisition, and maintenance projects;

- The formality and characteristics of project management techniques;
- The material changes that impact development, acquisition, and maintenance activities, such as:
 - Proposed or enacted changes in hardware, software, or vendors;
 - Proposed or enacted changes in business objectives or organizational structures; and
 - Proposed or enacted changes in key personnel positions.

Objective 2: Assess the level of oversight and support provided by the board and management relating to development, acquisition, and maintenance activities.

1. Assess the level of oversight and support by evaluating:
 - The alignment of business and technology objectives;
 - The frequency and quality of technology-related board reporting;
 - The commitment of the board and senior management to promote new products;
 - The level and quality of board-approved project standards and procedures;
 - The qualifications of technology managers; and
 - The sufficiency of technology budgets.

Objective 3: Assess the organizational structure in relation to the appropriateness of assigned responsibilities concerning technology systems and initiatives.

1. Evaluate organizational responsibilities to ensure the board and management:
 - Clearly define and appropriately assign responsibilities;
 - Appropriately assign security, audit, and quality assurance personnel to technology-related projects;
 - Establish appropriate segregation-of-duty or compensating controls; and
 - Establish appropriate project, technology committee, and board reporting requirements.

Objective 4: Assess the level and characteristics of risks associated with development, acquisition, and maintenance activities that could materially impact the organization.

1. Assess the risks identified in other objectives and evaluate the adequacy of risk management programs regarding:

- Risk identification and assessment procedures;
- Risk reporting and monitoring procedures; and
- Risk acceptance, mitigation, and transfer strategies.

Objective 5: Assess the adequacy of development project management standards, methodologies, and practices.

1. Evaluate the adequacy of development activities by assessing:

- The adequacy of, and adherence to, development standards and controls;
- The applicability and effectiveness of project management methodologies;
- The experience of project managers;
- The adequacy of project plans, particularly with regard to the inclusion of clearly defined:
 - Phase expectations;
 - Phase acceptance criteria;
 - Security and control requirements;
 - Testing requirements; and
 - Documentation requirements;
- The formality and effectiveness of quality assurance programs;
- The effectiveness of risk management programs;
- The adequacy of project request and approval procedures;
- The adequacy of feasibility studies;
- The adequacy of, and adherence to, standards and procedures relating to the:
 - Design phase;
 - Development phase;
 - Testing phase; and
 - Implementation phase;
- The adequacy of project change controls;
- The appropriate inclusion of organizational personnel throughout the project's life cycle;
- The effectiveness of project communication and reporting procedures; and
- The accuracy, effectiveness, and control of project management tools.

Objective 6: Assess the adequacy of acquisition project management standards, methodologies, and practices.

1. Assess the adequacy of acquisition activities by evaluating:
 - The adequacy of, and adherence to, acquisition standards and controls;
 - The applicability and effectiveness of project management methodologies;
 - The experience of project managers;
 - The adequacy of project plans, particularly with regard to the inclusion of clearly defined:
 - Phase expectations;
 - Phase acceptance criteria;
 - Security and control requirements; and
 - Testing, training, and implementation requirements;
 - The formality and effectiveness of quality assurance programs;
 - The effectiveness of risk management programs;
 - The adequacy of project request and approval procedures;
 - The adequacy of feasibility studies;
 - The adequacy of, and adherence to, standards that require request-for-proposals and invitations-to-tender to include:
 - Well-detailed security, reliability, and functionality specifications;
 - Well-defined performance and compatibility specifications; and
 - Well-defined design and development documentation requirements;
 - The adequacy of, and adherence to, standards that require:
 - Thorough reviews of vendors' financial condition and commitment to service; and
 - Thorough reviews of contracts and licensing agreements prior to signing;
 - The adequacy of contract and licensing provisions that address:
 - Performance assurances;
 - Software and data security provisions; and
 - Source-code accessibility/escrow assertions;
 - The adequacy of project change controls;
 - The appropriate inclusion of organizational personnel throughout the project's life cycle;
 - The effectiveness of project communication and reporting procedures; and
 - The accuracy, effectiveness, and control of project management tools.

Objective 7: Assess the adequacy of maintenance project management standards, methodologies, and practices.

1. Evaluate the sufficiency of, and adherence to, maintenance standards and controls relating to:
 - Change request and approval procedures;
 - Change testing procedures;
 - Change implementation procedures;
 - Change review procedures;
 - Change documentation procedures;
 - Change notification procedures
 - Library controls; and
 - Utility program controls.

Objective 8: Assess the effectiveness of conversion projects.

1. Evaluate the effectiveness of conversion projects by:
 - Comparing initial budgets and projected time lines against actual results;
 - Reviewing project management and technology committee reports;
 - Reviewing testing documentation and after-action reports;
 - Reviewing conversion after-action reports;
 - Interviewing technology and user personnel; and
 - Reviewing suspense accounts for outstanding items.

Objective 9: Assess the adequacy of quality assurance programs.

1. Assess the adequacy of quality assurance programs by evaluating:
 - The board's willingness to provide appropriate resources to quality assurance programs;
 - The completeness of quality assurance procedures (Are the deliverables of each project, and project phase, including the validation of initial project assumptions and approvals, appropriately assured?);
 - The scalability of quality assurance procedures (Are the procedures appropriately tailored to match the characteristics of the project?);
 - The measurability of quality assurance standards (Are deliverables assessed against predefined standards and expectations?);

- The adherence to problem-tracking standards that require:
 - Appropriate problem recordation;
 - Appropriate problem reporting;
 - Appropriate problem monitoring; and
 - Appropriate problem correction;
- The sufficiency of, and adherence to, testing standards that require:
 - The use of predefined, comprehensive test plans;
 - The involvement of end users;
 - The documentation of test results;
 - The prohibition against testing in production environments; and
 - The prohibition against testing with live data;
- The sufficiency and effectiveness of testing programs regarding:
 - The accuracy of programmed code;
 - The inclusion of expected functionality; and
 - The interoperability of applications and network components; and
- The independence of quality assurance personnel.

Objective 10: Assess the adequacy of program change controls.

1. Evaluate the sufficiency of, and adherence to:

- Routine and emergency program-change standards that require appropriate:
 - Request and approval procedures;
 - Testing procedures;
 - Implementation procedures;
 - Backup and backout procedures;
 - Documentation procedures; and
 - Notification procedures;
- Controls that restrict the unauthorized movement of programs or program modules/objects between development, testing, and production environments;
- Controls that restrict the unauthorized use of utility programs, such as:
 - Policy prohibitions;
 - Monitoring of use; and
 - Logical access controls;
- Library controls that restrict unauthorized access to programs outside an individual's assigned responsibilities such as:
 - Logical access controls on all libraries or objects within libraries; and

- Automated library controls that restrict library access and produce reports that identify who accessed a library, what was accessed, and what changes were made; and
- Version controls that facilitate the appropriate retention of programs, and program modules/objects, revisions, and documentation.

Objective 11: Assess the adequacy of patch-management standards and controls.

1. Evaluate the sufficiency of, and adherence to, patch-management standards and controls that require:
 - Detailed hardware and software inventories;
 - Patch identification procedures;
 - Patch evaluation procedures;
 - Patch request and approval procedures;
 - Patch testing procedures;
 - Backup and backout procedures;
 - Patch implementation procedures; and
 - Patch documentation.

Objective 12: Assess the quality of application, system, and project documentation, and the adequacy of documentation controls.

1. Assess the adequacy of documentation controls by evaluating the sufficiency of, and adherence to, documentation standards that require:
 - The assignment of documentation-custodian responsibilities;
 - The assignment of document authoring and approval responsibilities;
 - The establishment of standardized document formats; and
 - The establishment of appropriate documentation library and version controls.
2. Assess the quality of application documentation by evaluating the adequacy of internal and external assessments of:
 - Application design and coding standards;
 - Application descriptions;
 - Application design documents;
 - Application source-code listings (or in the case of object-oriented programming: object listings);

- Application routine naming conventions (or in the case of object-oriented programming: object naming conventions); and
 - Application operator instructions and user manuals.
3. Assess the quality of open source-code system documentation by evaluating the adequacy of internal and external assessments of:
- System design and coding standards;
 - System descriptions;
 - System design documents;
 - Source-code listings (or in the case of object-oriented programming: object listings);
 - Source-code routine naming conventions (or in the case of object-oriented programming: object naming conventions); and
 - System operation instructions.
4. Assess the quality of project documentation by evaluating the adequacy of documentation relating to the:
- Project request;
 - Feasibility study;
 - Initiation phase;
 - Planning phase;
 - Design phase;
 - Development phase;
 - Testing phase;
 - Implementation phase; and
 - Post-implementation reviews.

Note: If examiners employ sampling techniques, they should include planning and testing phase documentation in the sample.

Objective 13: Assess the security and integrity of system and application software.

1. Evaluate the security and integrity of system and application software by reviewing:
- The adequacy of quality assurance and testing programs;
 - The adequacy of security and internal-control design standards;
 - The adequacy of program change controls;

- The adequacy of involvement by audit and security personnel in software development and acquisition projects; and
- The adequacy of internal and external security and control audits.

Objective 14: Assess the ability of information technology solutions to meet the needs of the end users.

1. Interview end users to determine their assessment of technology solutions.

Objective 15: Assess the extent of end-user involvement in the system development and acquisition process.

1. Interview end users and review development and acquisition project documentation to determine the extent of end-user involvement.

CONCLUSIONS

Objective 16: Document and discuss findings and recommend corrective actions.

1. Document findings and recommendations regarding the quality and effectiveness of the organization's Development and Acquisition standards and procedures.
2. Discuss preliminary findings with the examiner-in-charge regarding:
 - Violations of laws, rulings, or regulations; and
 - Issues warranting inclusion in the report of examination.
3. Discuss your findings with management and obtain commitments for corrective actions and deadlines for remedying significant deficiencies.
4. Discuss findings with the examiner-in-charge regarding:
 - Recommendations regarding the Development and Acquisition rating; and
 - Recommendations regarding the impact of your conclusions on the composite rating(s).
5. Document your conclusions in a memo to the examiner-in-charge that provides report-ready comments for all relevant sections of the report of examination.
6. Organize your work papers to ensure clear support for significant findings and recommendations.

APPENDIX B: GLOSSARY

Application	A software program designed for use by end users.
Acceptance Criteria	Pre-established standards or requirements a product or project must meet.
Automated Controls	Software routines designed into programs to ensure the validity, accuracy, completeness, and availability of input, processed, and stored data.
Baseline	A documented version of a hardware component, software program, configuration, standard, procedure, or project management plan. Baseline versions are placed under formal change controls and should not be modified unless the changes are approved and documented.
Code	Software program instructions.
Database	An organized collection of information stored on one or more electronic files.
Deliverable	A project goal or expectation. Deliverables include broadly-defined, project or phase requirements and specifically-defined tasks within project phases.
Distributed Environment	A computer system with data and program components physically distributed across more than one computer.
End User	An individual who will utilize a product or program.
Enterprise Architecture	An organization's framework of technology hardware, software and related policies.
Flowcharts	Traditional flowcharts involve the use of geometric symbols, such as diamonds, ovals, and rectangles to represent the sequencing of program logic. Software packages are available that automatically chart programs or enable a programmer to chart a program without the need to draw it manually.
Functional Requirements	The business, operational, and security features an organization wants included in a program.
Iterative	Repetitive or cyclical. Iterative software development involves the completion of project tasks or phases in repetitive cycles. Tasks and phase activities are repeated until a desired result is achieved.

LAN	Local Area Network
Metrics	A quantitative measurement.
Milestone	Major project event.
Network	Two or more computer systems that are grouped together to share information, software, and hardware.
Object Code	Software program instructions compiled (translated) from source code into machine-readable formats.
Outsourcing	Contracting with third parties to perform activities, duties, or functions.
Operating System	Programs that collectively manage application programs. Operating systems allocate system resources, provide access and security controls, maintain file systems, and manage communications between end users and hardware devices.
Phase	A project segment.
Project	A task involving the acquisition, development or maintenance of a technology product.
Project Management	Planning, monitoring, and controlling an activity.
Script	Software program instructions.
Source Code	Software program instructions written in a format (language) readable by humans.
Spiral Development	An iterative project management model that focuses on the identification of project and product risks and the selection of project management techniques that best control the identified risks.
SDLC	Systems Development Life Cycle. A project management technique.