Department of Energy (DOE)

Software Engineering Methodology

March 1996

U. S. DEPARTMENT OF ENERGY

Assistant Secretary for Human Resources and Administration
Office of the Chief Information Officer

TITLE PAGE

Document Name:	Department of Energy Software Engineering Methodology
Publication Date:	March 1996
Contract Number:	DE-AC01-95AD34277
Project Number:	Task: 26-433-08-7-01 CDSI: 4001
Approval:	_____ Wayne Jones, HR-433

U.S. DEPARTMENT OF ENERGY

Assistant Secretary for Human Resources and Administration
Office of the Chief Information Officer

The compilation/development of the *Department of Energy Software Engineering Methodology* was performed as part of a continuing effort to improve the quality, performance, and productivity of Departmental information systems. Many Departmental and contractor personnel were involved in the development of the document as contributors or reviewers.

The following team of Departmental and contractor personnel served on the Steering Committee for the project. They generated the philosophical framework for the lifecycle methodology and provided oversight and guidance during the initial development process.

Paul Barkley
Brenda Coblentz
Roger Collins
Mark Gordy
Cheryle Greenaugh
William Greiner
Molly Jekel
Wayne Jones
John Kramer
Kathy McGlaughlin
Philip Quarry
Richard Tittle

The following individuals were the principal contributors to the document.

Andrew Bicocchi
Kathleen Canal
Denise Miller
Marilyn Morein
Margaret Webber

Credit must also be given to the many reviewers who provided invaluable input and feedback, and made useful suggestions for improving the content and presentation of information.

Paul Asbury
Lewis Bachman
Lorin Beehler
Carl Brown
Jo Buxton
Alison Campbell
Ann Heeney
Kim Hunter
Anne Imrie
Gail Kallmyer
William King
Milton Kowalewski
Robert Kuehne
Ron Lauzon
Fahy Low
Michael Mullan
Pam Pagnotta
Clif Quinn
Deborah Reynolds
Daniel Rupinski
Toni Russell
Pat A. Simmons
Sammy Slough
Ruth Ann Smith
Kimberly Wandersee
Harry Yocum

Chapter **Page**

1.0 Introduction 1.0-1
1.1 Implementation of Methodology 1.1-1
1.2 Submitting Change Requests 1.2-1

2.0 Lifecycle Model 2.0-1
2.1 Project Sizes 2.1-1
2.2 Adapting the Lifecycle 2.2-1
2.3 Development Methodologies 2.3-1

3.0 Planning Stage 3.0-1
3.1 Analyze User Environment 3.1-1
3.2 Define Project Objectives 3.2-1
3.3 Define Project Scope 3.3-1
3.4 Develop High-Level Project Requirements 3.4-1
3.5 Establish Communication With Functional Areas 3.5-1
3.6 Determine Project Feasibility 3.6-1
3.6.1 Investigate Software Alternatives 3.6-3
3.6.2 Investigate Hardware Alternatives 3.6-7
3.6.3 Formulate Platform Options 3.6-8
3.6.4 Conduct Feasibility Review 3.6-9
3.6.5 Conduct Analysis of Benefits and Costs 3.6-11
3.6.6 Conduct Feasibility Study 3.6-12
3.6.6.1 Analyze the alternatives 3.6-14
3.6.6.2 Determine feasibility recommendations 3.6-15
3.6.6.3 Develop feasibility study document 3.6-16
3.7 Develop Project Plan 3.7-1
3.8 Develop Software Quality Assurance Plan 3.8-1
3.9 Conduct In-Stage Assessment 3.9-1
3.10 Conduct Planning Stage Exit 3.10-1

4.0 Requirements Definition Stage 4.0-1
4.1 Develop Software Configuration Management Plan 4.1-1
4.2 Select Requirements Analysis Methodology 4.2-1
4.3 Define Project Requirements 4.3-1
4.3.1 Define Functional Requirements 4.3-3
4.3.2 Define Input and Output Requirements 4.3-5
4.3.3 Define Performance Requirements 4.3-6
4.3.4 Define User Interface Requirements 4.3-7
4.3.5 Define System Interface Requirements 4.3-8
4.3.6 Define Communication Requirements 4.3-9
4.3.7 Define Computer Security and Access Requirements 4.3-10
4.3.8 Define Backup and Recovery Requirements 4.3-13
4.3.9 Define Data Requirements 4.3-16

Chapter **Page**

4.3.10 Define Implementation Requirements 4.3-17
4.4 Compile Project Requirements 4.4-1
4.4.1 Develop Requirements Traceability Matrix 4.4-2
4.4.2 Develop Software Requirements Specification 4.4-4
4.5 Establish Functional Baseline 4.5-1
4.6 Develop Project Test Plan 4.6-1
4.6.1 Identify Test Methodologies 4.6-3
4.6.2 Identify Test Phases 4.6-4
4.6.3 Identify Test Environment Requirements 4.6-6
4.7 Develop Acceptance Test Plan 4.7-1
4.8 Select Design Methodology 4.8-1
4.9 Revise Project Plan 4.9-1
4.10 Conduct In-Stage Assessment 4.10-1
4.11 Conduct Requirements Definition Stage Exit 4.11-1

5.0 Functional Design Stage 5.0-1
5.1 Determine Software Structure 5.1-1
5.1.1 Identify Design Entities 5.1-2
5.1.2 Identify Design Dependencies 5.1-4
5.2 Design Content of System Inputs and Outputs 5.2-1
5.3 Design User Interface 5.3-1
5.3.1 Design Menu Hierarchy 5.3-3
5.3.2 Design Data Entry Screens 5.3-5
5.3.3 Design Display Screens 5.3-6
5.3.4 Design Online Help 5.3-8
5.3.5 Design System Messages 5.3-10
5.4 Design System Interfaces 5.4-1
5.5 Design System Security Controls 5.5-1
5.6 Build Logical Model 5.6-1
5.7 Build Data Model 5.7-1
5.8 Develop Functional Design 5.8-1
5.8.1 Develop Functional Design Document 5.8-2
5.8.2 Conduct Functional Design Review 5.8-3
5.9 Initiate Procurement of Hardware and Software 5.9-1
5.10 Revise Project Plan 5.10-1
5.11 Conduct In-Stage Assessment 5.11-1
5.12 Conduct Functional Design Stage Exit 5.12-1

Chapter **Page**

6.0 System Design Stage 6.0-1
6.1 Select System Architecture 6.1-1
6.1.1 Evaluate System Architecture Alternatives 6.1-2
6.1.2 Recommend System Architecture 6.1-5
6.2 Design Specifications for Software Modules 6.2-1
6.3 Design Physical Model and Data Base Structure 6.3-1
6.4 Develop Integration Test Plan 6.4-1
6.5 Develop System Test Plan 6.5-1
6.6 Develop Conversion Plan 6.6-1
6.7 Develop System Design 6.7-1
6.7.1 Develop System Design Document 6.7-2
6.7.2 Conduct Critical Design Review 6.7-3
6.8 Develop Program Specifications 6.8-1
6.9 Define Programming Standards 6.9-1
6.10 Revise Project Plan 6.10-1
6.11 Conduct In-Stage Assessment 6.11-1
6.12 Conduct System Design Stage Exit 6.12-1

7.0 Programming Stage 7.0-1
7.1 Develop Acquisition Plan 7.1-1
7.2 Develop Installation Plan 7.2-1
7.3 Establish Programming Environment 7.3.1
7.4 Write Programs 7.4-1
7.5 Conduct Unit Testing 7.5-1
7.6 Establish Development Baselines 7.6-1
7.7 Plan Transition to Operational Status 7.7-1
7.8 Generate Operating Documentation 7.8-1
7.8.1 Produce Users Manual 7.8-3
7.8.2 Produce Programmers Reference Manual 7.8-5
7.9 Develop Training Program 7.9-1
7.10 Revise Project Plan 7.10-1
7.11 Conduct In-Stage Assessment 7.11-1
7.12 Conduct Programming Stage Exit 7.12-1

8.0 Software Integration and Testing Stage 8.0-1
8.1 Conduct Integration Testing 8.1-1
8.2 Conduct System Testing 8.2-1
8.3 Initiate Acceptance Process 8.3-1
8.4 Conduct Acceptance Test Team Training 8.4-1
8.5 Revise Project Plan 8.5-1
8.6 Conduct In-Stage Assessment 8.6-1
8.7 Conduct Software Integration and Testing Stage Exit 8.7-1
9.0 Installation and Acceptance Stage 9.0-1

Chapter **Page**

9.1 Perform Installation Activities 9.1-1
9.2 Conduct Installation Tests 9.2-1
9.3 Conduct User Training 9.3-1
9.4 Conduct Acceptance Test 9.4-1
9.5 Conclude Acceptance Process 9.5-1
9.6 Transition to Operational Status 9.6-1
9.7 Revise Project Plan 9.7-1
9.8 Conduct In-Stage Assessment 9.8-1
9.9 Conduct Installation and Acceptance Stage Exit 9.9-1

10.0 Software Maintenance 10.0-1
10.1 Problem/Modification Identification Stage 10.1-1
10.2 Analysis Stage 10.2-1
10.3 Design Stage 10.3-1
10.4 Programming Stage 10.4-1
10.5 System Test Stage 10.5-1
10.6 Acceptance Stage 10.6-1
10.7 Delivery Stage 10.7-1

Appendix

A Glossary

B List of Abbreviations

C Conducting Structured Walkthroughs

D In-Stage Assessment Process Guide

E Stage Exit Process Guide

F Computer System Retirement Guidelines

G Site-Specific Information Systems Environment

Software Engineering Methodology

Chapter 1.0
Introduction

Table of Contents

Chapter **Page**

1.0 Introduction 1.0-1
1.1 Implementation of Methodology 1.1-1
1.2 Submitting Change Requests 1.2-1

Chapter: **1.0**
Introduction

Description: The *Department of Energy Software Engineering Methodology* provides guidance for software engineering, project management, and quality assurance practices and procedures. The primary purpose of the methodology is to promote the development of reliable, cost-effective, computer-based software products while making efficient use of resources. Use of the methodology will also aid in the status tracking, management control, and documentation efforts of the project.

This software engineering methodology is consistent with other methodologies used in the Government and private industry. It complies with Departmental policy on project management, configuration management, security, and records management. Significant input for the methodology was obtained from software management programs at sites and organizations throughout the Department. The methodology integrates Departmental best practices and focuses on the quality of both the software engineering process and the work products generated from the process.

The software engineering methodology is derived from the principles and standards advocated by software quality industry leaders, such as The Institute of Electrical and Electronics Engineers and the Carnegie-Mellon Software Engineering Institute (SEI). This methodology is designed to enable project teams to fully achieve Level 2 maturity on the SEI Capability Maturity Model. Some Level 3 key process areas are also incorporated into the methodology.

Software quality assurance is integrated into the software engineering methodology, making quality the responsibility of all project team manager(s) and members. To assure the development of quality software products, the methodology prescribes reviews, inspections, and audits for the lifecycle processes and technical work products. To protect the integrity of the software, the methodology also prescribes configuration controls over software, data, and technical documentation.

The software engineering methodology encompasses all aspects of the software lifecycle from project planning through production and maintenance, and integrates the following basic lifecycle management concepts.

- Implementation of software engineering preferred practices using a graded approach based on the level of effort, complexity, and degree of external impact of the software product.

Description,

continued:

- Implementation of a project management methodology including quality assurance, configuration management, and a comprehensive testing approach that is adaptable to the individual site environments.

- Application of a complete documentation approach supporting both lifecycle and project management activities, to assure an effective method for managing, tracking, and evaluating software engineering activities.

The software engineering methodology is intended to be used by individuals, project teams, and managers who are responsible for developing a new computer-based software product or make changes to an existing system. The methodology will be reviewed on a regular basis and modified as needed to keep pace with the changing needs of the Departmental software engineering environment and the continuing technical advances in the information systems industry.

The following sections provide additional information about using this software engineering methodology.

1.1 Implementation of Methodology
1.2 Submitting Change Requests

Section: **1.1**
Implementation of Methodology

Description: This methodology integrates software engineering, project management, and quality assurance practices and is designed to be flexible. It can be adapted to accommodate the specific needs of any software project and all computing platforms used in the Department including standalone and networked mainframes, minicomputers, and microcomputers.

Projects that were initiated prior to the publication of this document should plan to implement the methodology at the earliest feasible stage or the next release of the product. If a Project Plan already exists, make the revisions necessary to integrate the software engineering, project management, and quality assurance practices, as appropriate. If a Project Plan does not exist, develop a plan that summarizes the activities and deliverables of the previous stages and incorporates the methodology activities and products into the subsequent stages.

The software engineering methodology presented here does not supersede, replace, or override more stringent requirements that may apply to specific projects such as scientific and technical practices, and security and safety issues associated with the Nuclear Weapons Complex.

Since the methodology does not provide specific guidance for every software engineering situation, suggestions for adapting the methodology to accommodate projects of varying size, complexity, or criticality are provided in *Chapter 2.0, Lifecycle Model.*

Questions: If specific questions are generated concerning the interpretation or applicability of portions of the methodology, the project team should attempt to resolve them during the project review activities built into the stages of the lifecycle. The system owner/user(s) and other project stakeholders must concur with any adaptations that are made.

When questions about interpretation or applicability of the guidance to a specific project cannot be resolved by the project team, the issue should be submitted to the site authority for software engineering, such as the site Information Resources Management or Information Management organization, for resolution. For Headquarters projects, the Office of Information Management is the site authority.

Questions, continued:

Questions and issues will be analyzed by the site Information Management Division and a response will be made in one of the following ways.

- An immediate solution is determined and will be provided to the project team.

- The issue will be submitted to personnel who are considered experts in the area in question. Once a solution is reached, it will be provided to the project team.

It is important to also submit questions of interpretation or applicability and the site-specific resolution to the Headquarters Office of Information Management. The Office will determine if a modification to the *Software Engineering Methodology* is needed to clarify processes or to provide additional adaptation suggestions. A central clearinghouse for all questions and resolutions will ensure that needed changes to the methodology can be identified and implemented in a timely and consistent manner.

Section: **1.2**
Submitting Change Requests

Description: The Departmental software engineering environment is continuously changing as emerging technologies are integrated into projects, system owner/user requirements are expanded, and organizational needs evolve. The software engineering methodology will be expanded and revised, as needed, to reflect changes in the environment, improvements suggested through user feedback, and the maturation of software engineering capabilities.

Users of the methodology are encouraged to submit suggestions for improving its content and to report any practices that are difficult to understand or create an implementation problem for a project team.

Suggestions and problems should be submitted on the Change Request Form that is provided at the end of this section. If the form is not available or does not accommodate the type of request being made, submit a memo that describes the suggestion or problem.

The Change Request Form or memo should be submitted to the Office of Information Management at Headquarters in Germantown, Maryland. The Change Request should be submitted through the site's Information Management organization. All requests will be evaluated and the originator of the request will be notified of the action taken.

Some requests will be handled immediately while others may require investigation by an ad hoc Working Group of knowledgeable personnel. In some cases, a request may not be appropriate for the current environment, but will be retained for future consideration.

Software Engineering Methodology
Change Request Form

To be completed by Requestor	To be completed by Analyst
Name:	Name:
Phone:	Phone:
Location:	Location:
Date:	Date Assigned:
Document Section:	Request Number:
Requested Change and Justification: Check if additional pages are attached ___	Change Classification Data (check one) Class I Change ___ Class II Change ___ Class III Change ___ Summary of Impact: Check if additional pages are attached ___
Change Impacts Section Number: _____ Pages: _____	Change Impacts Section Number: _____ Pages: _____

Working Group Actions

Approval/Disapproval Reason(s): Date _____

Additional Comments:

Software Engineering Methodology
Change Request Instructions

Requestor's Section	Instructions
Name: Phone: Location: Date:	Fill in your name, telephone number, location and date.
Document Section:	List the document section where you want to make a change.
Requested Change and Justification:	State the change that you want to incorporate and state your reasons for the change. Attach additional pages as needed.
Change Impacts:	List any section numbers and pages that will be affected by the proposed change.
**Send the completed Change Request Form to the Headquarters Office of Information Management, Systems Engineering Group, Engineering Services Team, HR-433, Germantown, Maryland.	

Analyst's Section	Instructions
Name: Phone: Location: Date Assigned:	Fill in your name, telephone number, location, and date you received the assignment.
Request Number:	Obtain the sequential number that will be used to track the request from the Headquarters Office of Information Management, Systems Engineering Group, Engineering Services Team, HR-433.
Change Classification Data:	**Class I Changes** in policy, procedures, required actions, or deliverables are defined by Government units (Congress, Office of Management and Budget), or by DOE policies, procedures, and administrative requirements. These changes must be reviewed and approved by HR-433 and incorporated into the methodology with the next update.
	Class II Changes in technology or development methodology are descriptions of innovations in the way software products are developed. These changes require review and concurrence by a Working Group and must be approved by HR-433. Changes in required deliverables may be implemented by HR-433 or recommended by a Working Group. The impact on the remainder of the methodology when such changes are incorporated require review and concurrence by a Working Group and approval by HR-433.
	Class III Changes in factual information (security requirements), wording, or corrections of typographical errors will be implemented as soon as possible to keep the methodology accurate and current. Corrections of typographical errors are implemented without review. All other changes require review and approval by HR-433.
Summary of Impact:	State what effect the proposed change would have on other sections of the document or the methodology.
Change Impacts:	List any section numbers and pages that will be affected by the proposed change.

Working Group	Instructions
Approval/Disapproval Reasons:	State whether the proposed change should be approved or disapproved. Give reasons for the decision. Indicate date of approval/disapproval.

Software Engineering Methodology

Chapter 2.0
Lifecycle Model

Table of Contents

Chapter **Page**

2.0 Lifecycle Model 2.0-1
- 2.1 Project Sizes 2.1-1
- 2.2 Adapting the Lifecycle 2.2-1
- 2.3 Development Methodologies 2.3-1

Chapter: **2.0**
Lifecycle Model

Description: This chapter describes the lifecycle model used for the Departmental software engineering methodology. This model partitions the software engineering lifecycle into eight major stages, as shown in *Exhibit 2.0-1, Software Lifecycle Stages and Deliverables.* Each stage is divided into activities and tasks, and has a measurable end point (Stage Exit). The execution of all eight stages is based on the premise that the quality and success of a software product depends on a feasible concept, highly visible project planning, commitments to resources and schedules, complete and accurate requirements, a sound design, consistent and maintainable programming techniques, and a comprehensive testing program. The lifecycle stages and activities are described in chapters 3.0 through 10.0.

Intermediate work products are produced during the performance of the activities and tasks in each stage. These work products are inspected and can be used to assess software integrity, quality, and project status. As a result, adequacy of requirements, correctness of designs, and quality of software products become known early in the effort.

At least one time during each stage, an In-Stage Assessment is performed. An In-Stage assessment is an independent review of the work products and deliverables developed or revised during each lifecycle stage. The assessment is typically conducted by a Quality Assurance practitioner and the results are provided to the project manager. In-Stage Assessments are recommended after the achievement of all major project milestones and the completion of deliverable work products.

At the conclusion of each stage, a Stage Exit is initiated to review the work products of that stage and to determine whether to proceed to the next stage, continue work in the current stage, or abandon the project. The approval of the system owner and other project stakeholders at the conclusion of each stage enables both the system owner and the project manager to remain in control of the project throughout its life, and prevents the project from proceeding beyond authorized milestones.

The end products of the lifecycle are the software product, the data managed by the software, associated technical documentation, and user training and support. The end products and services are maintained throughout the remainder of the lifecycle in accordance with documented configuration management procedures.

Description, continued:

The lifecycle model provides a method for performing the individual activities and tasks within an overall project framework. The stages and activities are designed to follow each other in an integrated fashion. Project teams have the flexibility to adapt the lifecycle model to accommodate a particular development methodology (e.g., spiral development), software engineering technique (e.g., prototyping and rapid application development), or other project constraints.

The amount of project and system documentation required throughout the lifecycle depends on the size and scope of the project. System documentation needs to be at a level that allows for full system operability, usability, and maintainability. Typically, projects that require at least one work-year of effort should have a full complement of documentation. For projects that require less than one work-year of effort, the project manager and system owner should determine the documentation requirements. In addition, the project's security and quality assurance criteria may require the performance of other activities and the generation of additional documentation.

The requirements for documentation should not be interpreted as mandating formal, standalone, printed documents in all cases. Progressive documents that continuously revise and expand existing documentation, online documents, forms, reports, electronic mail messages, and handwritten notes (e.g., informal conference records) are some examples of alternative documentation formats.

The following sections provide additional information about the lifecycle model.

2.1 Project Sizes
2.2 Adapting the Lifecycle
2.3 Development Methodologies

Exhibit 2.0-1. Software Lifecycle Stages and Deliverables

Planning	Feasibility Statement Project Plan Software Quality Assurance Plan
Requirements Definition	Software Configuration Management Plan Continuity of Operations Statement/Plan Software Requirements Specification Project Test Plan Acceptance Test Plan (draft)
Functional Design	Logical Model Data Dictionary Requirements Traceability Matrix Functional Design Document
System Design	Physical Model Integration Test Plan (draft) System Test Plan (draft) Conversion Plan System Design Document Program Specifications Programming Standards
Programming	Acquisition Plan Installation Plan (draft) Integration Test Plan (final) System Test Plan (final) Software Baseline Transition Plan Operating Documents (draft) Training Plan (draft)
Software Integration & Testing	Integration Test Reports System Test Report Operating Documents (final) Training Plan (final) Installation Plan (final) Acceptance Test Plan (final) Preacceptance Checklist

Installation & Acceptance

User Training Materials
Acceptance Test Report
Acceptance Checklist
Operational System

Software Maintenance

Section: **2.1**
Project Sizes

Description: The lifecycle model used in this software engineering methodology can be applied to software projects of varying sizes. In this model, software projects are divided into three sizes: large, medium, and small. Each project size uses the same lifecycle stages. Medium and small projects may compress or combine stages and required documentation in direct proportion to the size of the development effort. The major differences between project sizes are determined by the following items.

- The estimated total labor hours (the level of effort) required to complete the project.
- The use of cutting edge or existing technology.
- The type and extent of both user and system interface requirements.
- The project's contribution to, and impact on, the activities carried out by the system users and other Departmental organizations.

The requirements, constraints, and risks associated with the project also influence the determination of project size. The project size and any plans for adapting the lifecycle model are documented in the Project Plan, which is reviewed and approved by the system and other project stakeholders.

The following subsections provide descriptions of the three project sizes used in this lifecycle model. *Exhibit 2.1-1, Software Project Sizes,* shows the level of effort and complexity measures used to define the three sizes.

Large Projects: Large software engineering projects are included in the system owner's organizational long-range plans. Headquarters-wide and Departmentwide projects are usually developed as large-sized projects and are likely to require a major acquisition of hardware and software. Typically, the larger the size and scope of the project, the greater the detail and coordination needed to manage the project. As risk factors and levels of effort increase, the scope of project management also increases and becomes a critical factor in the success of the project.

Medium Projects: Medium software engineering projects require less effort than large projects, typically use existing hardware and software, and might not be captured during the organizational long-range planning process. Medium size projects are frequently developed to automate operations within a programmatic office or among a limited number of sites, and may be used to interface with other software products. Planning medium size projects within the context of the system owner organization's overall mission, and building in compatibility to the Departmental computing environment can improve the software product's ability to interface with other users, organizations, and applications; and increase the product's longevity.

Small Projects: Small software engineering projects require minimal effort and use existing hardware and software. The operational details of a small project can easily be managed by the project manager, so formal documentation requirements are limited. A project is small when the software being developed will have limited functionality and use, meets a one-time requirement, or is developed using reusable code.

Exhibit 2.1-1. Software Project Sizes

Complexity (and associated characteristics)	**Effort Required** (in staff months)		
	0-8	**9-24**	**25-n**
Low: - Existing or known technology - Simple interfaces - Requirements well known - Skills are available	Small	Small	Medium
Medium: - Some new technology - Multiple interfaces - Requirements not well known - Skills not readily available	Small	Medium	Large
High: - New technology - Numerous complex interfaces - Numerous resources required - Skills must be acquired	Medium	Large	Large

Section: **2.2**
Adapting the Lifecycle

Description: The software engineering methodology implements well-defined processes in a lifecycle model that can be adapted to meet the specific requirements or constraints of any software project. This section provides guidelines for adapting the lifecycle processes to fit the characteristics of the project. These guidelines help ensure that there is a common basis across all software projects for planning, implementing, tracking, and assuring the quality of the work products.

The lifecycle model has built-in flexibility. All of the stages and activities can be adapted to any size and scope software engineering project. The lifecycle can be successfully applied to software development projects, software maintenance or enhancements, and customization of commercial software. The lifecycle is appropriate for all types of administrative, business, manufacturing, laboratory, scientific, and technical applications. For scientific and technical projects, adaptations to the lifecycle may be dictated by the project stakeholders or the requirements for reporting technical results in formal reports or journal articles.

Adaptations: The lifecycle can be compressed to satisfy the needs of a small project, expanded to include additional activities or work products for a large or complex project, or supplemented to accommodate security requirements. Any modifications to the lifecycle should be consistent with the established activities, documentation, and quality standards included in the methodology. Project teams are encouraged to adapt the lifecycle as long as the fundamental software engineering objectives are retained and quality is not compromised.

The following are some examples of lifecycle adaptations.

- Change the order in which lifecycle stages are performed.
- Schedule stages and activities in concurrent or sequential order.
- Repeat, merge, or eliminate stages, activities, or work products.
- Include additional activities, tasks, or work products in a stage.
- Change the sequence or implementation of lifecycle activities.
- Change the development schedule of the work products.
- Combine or expand activities and the timing of their execution.

Adaptations, continued:

The lifecycle forms the foundation for project planning, scheduling, risk management, and estimation. When a lifecycle stage, activity, or work product is adapted, the change must be identified, described, and justified in the Project Plan. *Exhibit 2.2-1, Adapting the Lifecycle,* shows how stages can be combined to accommodate different size projects and software engineering techniques. *Notes* are provided throughout the lifecycle stage chapters to identify activities that have built-in project adaptation strategies. Adaptations should not introduce an unacceptable level of risk and require the approval of the system owner and other project stakeholders.

When adapting the lifecycle model, care must be taken to avoid the following pitfalls.

- Incomplete and inadequate project planning.
- Incomplete and inadequate definition of project objectives and software requirements.
- Lack of a development methodology that is supported by software engineering preferred practices and tools.
- Insufficient time allocated to complete design before coding is started.
- Not defining and meeting criteria for completing one software lifecycle stage before beginning the next.
- Compressing or eliminating testing activities to maintain an unrealistic schedule.

Sample Statements:

The following are sample statements that can be used in the Project Plan to describe different types of lifecycle adaptations. The first example shows a scenario where the Feasibility Study activity will not be conducted in the Planning Stage.

A Feasibility Study will not be performed for this software project. The need for the product has been documented in several organizational reports and was included in the fiscal year long-range plans. The platform for the project is currently used for all applications owned by this organization. There are no known vendor packages that will satisfy the functional requirements described by the system owner.

Sample

Statements, continued:

The following is a sample statement that shows how work products from two different stages can be combined into one deliverable.

The Functional Design and System Design documents will be combined into one design document. A Stage Exit will be conducted when the design document is completed. To reduce the risk associated with combining the two documents, the project team will develop prototype screens and reports for review and approval by the system owner/user(s) as the prototypes are developed.

The following is a sample that shows how the eight lifecycle stages can be compressed into five stages for a small project.

This project will require 4 staff months of effort to enhance an existing application. The eight stages in the lifecycle will be combined into five stages as follows: (1) Planning, (2) Requirements and Design, (3) Programming and Testing, (4) Installation and Acceptance, and (5) Maintenance.

The following deviations will occur for document deliverables:

- *A Feasibility Study and an Analysis of Benefits and Costs will not be necessary due to the restricted software and hardware platform.*
- *The Requirements Specification will be limited to the statement of enhancement requirements.*
- *The Functional Design and System Design documents will be combined into one design document.*
- *An amendment package will be developed for the existing Users Manual.*

Exhibit 2.2-1. Adapting the Lifecycle

```
PROJECT
 SIZE
                          ▼         ▼         ▼         ▼         ▼         ▼         ▼
                         *         *         *         *         *         *         *         *
LARGE                    /))))))))3))))))))3))))))))3))))))))3))))))))3))))))))3))))))))1
                         Maintenance
                         *Planning *Req.Def. *Fun.Des. *Sys.Des. *Progrmg. *Testing  *Install. * &
Ops.
                                                   +)))))))))))))),           Accpt.
                                                   *ITERATIVE DEV. /)),(1)
                                              +))3))))))))))))))1◂)-
                                              .)▸*next function  *
                                                   .))))))))))))))-

                          ▼                   ▼                   ▼                   ▼

                         *         *                   *                   *                   *
MEDIUM                   /))))))))3))))))))))))))3))))))))))))))))3))))))0))))))1  Maintenance
                         *Planning *Req./Fun.Des. *Sys.Des./Progrm. *Test.  *Instl. * & Ops.
                                    +))))))))))))),                        Accpt.
                                +)▸*RAPID PROTO. /)),(2)
                                .))2))))))))))))-◂)-

                                              ▼                         ▼

                         *                    *                         *
SMALL                    /))))))))0))))))))))))3))))))))))))))0))))))))))1  Maintenance
                         *Planning *Req./Desg.  *Progr./Test.  *Install.  * & Ops.
                         R         R            R              RAccept.   R
```

```
                    LESS   ))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))Q ►
MORE
                                 DEGREE OF PROJECT MANAGEMENT REQUIRED
```

Note: Iterative development and rapid prototyping are optional techniques that can be used on any size project.

▾ = Stage exit occurs at this point.

(1) Each iteration produces working function(s) from integrated program modules.
(2) May produce any or all of requirements, system architecture, system design.

Section: **2.3**
Development Methodologies

Description: This section describes some examples of development methodologies and techniques that can be used with the Departmental software engineering methodology. The examples include high-level instructions on how to adapt the lifecycle stages to accommodate the development methodology. *Exhibit 2.2-1, Adapting the Lifecycle,* shows how some development methodologies and techniques can be used with the lifecycle model. The examples provided here are not intended to be a comprehensive list of development methodologies and techniques.

Segmented Development: Segmented development is most often applied to large software engineering projects where the project requirements can be divided into functional segments. Each segment becomes a separate project and provides a useful subset of the total capabilities of the full product. This segmentation serves two purposes: to break a large development effort into manageable pieces for easier project management and control; and to provide intermediate work products that form the building blocks for the complete product.

The lifecycle processes and activities are applied to each segment. The overall system and software objectives are defined, the system architecture is selected for the overall project, and a Project Plan for development of the first segment is written and approved by the system owner.

Segments are delivered to the system owner for evaluation or actual operation. The results of the evaluation or operation are then used to refine the content of the next segment. The next segment provides additional capabilities. This process is repeated until the entire software product has been developed. If significant problems are encountered with a segment, it may be necessary to reexamine and revise the project objectives, modify the system architecture, update the overall schedule, or change how the segments are divided.

Two major advantages of this approach are: the project manager can demonstrate concrete evidence that the final product will work as specified; and users will have access to, and use of, segments or functions prior to the delivery of the entire software product.

Spiral

Development: Spiral development repeats the planning, requirements, and functional design stages in a succession of cycles in which the project's objectives are clarified, alternatives are defined, risks and constraints are identified, and a prototype is constructed. The prototype is evaluated and the next cycle is planned.

The project objectives, alternatives, constraints, and risks are refined based on this evaluation; then, an improved prototype is constructed. This process of refinement and prototyping is repeated as many times as necessary to provide an incrementally firm foundation on which to proceed with the project.

The lifecycle activities for the Planning, Requirements Definition, and Functional Design Stages are repeated in each cycle. Once the design is firm, the lifecycle stages for System Design, Programming, and Integration and Testing are followed to produce the final software product.

Rapid Prototyping: Rapid prototyping can be applied to any software development methodology (e.g., segmented, spiral). Rapid prototyping is recommended for software development that is based on a new technology or evolutionary requirements.

With the rapid prototyping technique, the most important and critical software requirements are defined based on current knowledge and experience. A quick design addressing those requirements is prepared, and a prototype is coded and tested. The purpose of the prototype is to gain preliminary information about the total requirements and confidence in the correctness of the design approach. Characteristics needed in the final software product, such as efficiency, maintainability, capacity, and adaptability might be ignored in the prototype.

The prototype is evaluated, preferably with extensive user participation, to refine the initial requirements and design. After confidence in the requirements and design approach is achieved, the final software is developed. The prototype might be discarded, or a portion of it used to develop the final product.

The normal software engineering documentation requirements are usually postponed with prototyping efforts. Typically, the project team, project stakeholders, and system owner agree that the prototype will be replaced with the actual software product and required support documentation after proof of the model. The software that replaces the prototype should be developed using the lifecycle processes and activities.

Iterative Technique:

The iterative technique is normally used to develop software products piece by piece. Once the system architecture and functional or conceptual design are defined and approved, system functionality can be divided into logically related pieces called "drivers."

In iterative fashion, the project team performs system design, code, unit test, and integration test activities for each driver, thereby delivering a working function of the product. These working functions or pieces of the software product are designed to fit together as they are developed. This technique allows functions to be delivered incrementally for testing so that they can work in parallel with the project team. It also enables other functional areas, such as documentation and training, to begin performing their activities earlier and in a more parallel effort. In addition, the iterative technique enables progress to be visible earlier, and problems to be contained to a smaller scope.

With each iterative step of the development effort, the project team performs the lifecycle processes and activities.

Software Engineering Methodology

Chapter 3.0
Planning Stage

Table of Contents

Chapter **Page**

3.0 Planning Stage 3.0-1
- 3.1 Analyze User Environment 3.1-1
- 3.2 Define Project Objectives 3.2-1
- 3.3 Define Project Scope 3.3-1
- 3.4 Develop High-Level Project Requirements 3.4-1
- 3.5 Establish Communication With Functional Areas 3.5-1
- 3.6 Determine Project Feasibility 3.6-1
  - 3.6.1 Investigate Software Alternatives 3.6-3
  - 3.6.2 Investigate Hardware Alternatives 3.6-7
  - 3.6.3 Formulate Platform Options 3.6-8
  - 3.6.4 Conduct Feasibility Review 3.6-9
  - 3.6.5 Conduct Analysis of Benefits and Costs 3.6-11
  - 3.6.6 Conduct Feasibility Study 3.6-12
    - 3.6.6.1 Analyze the alternatives 3.6-14
    - 3.6.6.2 Determine feasibility recommendations 3.6-15
    - 3.6.6.3 Develop feasibility study document 3.6-16
- 3.7 Develop Project Plan 3.7-1
- 3.8 Develop Software Quality Assurance Plan 3.8-1
- 3.9 Conduct In-Stage Assessment 3.9-1
- 3.10 Conduct Planning Stage Exit 3.10-1

Chapter: **3.0**
Planning Stage

Description: This is the first stage in the lifecycle of a software engineering project. In this stage, the users' environment is analyzed, the project objectives and scope are defined, the high-level requirements are estimated, the feasibility of the project is determined, and the initial Project Plan is developed and approved.

This stage is initiated when a project manager/team responds to a DOE-approved task assignment with a Management Plan and related materials. The preparation of the Management Plan and related materials involves several critical planning issues such as the identification of preliminary requirements; staff, schedule, and cost estimates; and potential risks associated with the project. This information is reviewed in the Planning Stage and forms the foundation for all subsequent planning activities.

During this stage, the system owner and users are interviewed to: identify their business needs and expectations for the product; gain a common understanding of the task assignment; and determine how the project supports the DOE and organizational missions and long-range information resource management plans. The system owner is the organizational unit that is funding the project, and users are the DOE employees and contractors who will use the product.

In this stage, the project team should be focused on identifying what the project will automate, and whether developing an automated solution makes sense from business, cost, and technical perspectives. If the project is feasible, time, cost, and resource estimates must be formulated for the project, and risk factors must be assessed. It is important for the project team to work closely with representatives from all functional areas that will be involved in providing resources, information, or support services for the project. The information gathered in this stage is used to plan and manage the project throughout its lifecycle.

Input: The following items provide input to this stage.

- Task Assignment (Statement of Work)
- Management Plan and related materials

High-Level Activities:

The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage. These activities represent the minimum requirements for a large software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

3.1 Analyze User Environment
3.2 Define Project Objectives
3.3 Define Project Scope
3.4 Develop High-Level Project Requirements
3.5 Establish Communications With Functional Areas
3.6 Determine Project Feasibility
3.7 Develop Project Plan
3.8 Develop Software Quality Assurance Plan
3.9 Conduct In-Stage Assessment
3.10 Conduct Planning Stage Exit

Output:

Several work products are developed during this stage. The work products listed below are the minimum requirements for a large software project. Deviations in the content and delivery of these work products are determined by the size and complexity of a project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

- Project File
- Description of user environment
- Statement of project objectives
- Statement of project scope
- Statement of high-level project requirements
- Functional area contact list and project profile
- Summary of platform options
- Statement of project feasibility
- Analysis of Benefits and Costs Report
- Feasibility Study Document
- Project Plan
- Software Quality Assurance Plan

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 3.0-1, Planning Stage Activities and Work Products by Project Size*. The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large projects.

Review Process: Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. The time and resources needed to conduct the walkthroughs should be reflected in the project resources, schedule, and work breakdown structure.

Reference: *Appendix C, Conducting Structured Walkthroughs*, provides a procedure and sample forms that can be used for structured walkthroughs.

Bibliography: The following materials were used in the preparation of the Planning Stage chapter.

1. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes*, IEEE Std 1074-1991, New York, 1992.

2. U.S. Department of Energy, *Analysis of Benefits and Costs (ABC's) Guideline: Volume 1, A Manager's Guide to Analysis of Benefits and Costs*, DOE/MA-0342, June 1988.

3. U.S. Department of Energy, *Analysis of Benefits and Costs (ABC's) Guideline: Volume 2, An Analyst's Handbook for Analysis of Benefits and Costs*, DOE/MA-0343, June 1988.

4. U.S. Department of Energy, *Software Management Guide*, DOE/AD-0028, 1992.

5. U.S. Department of Energy, *DOE/NV Software Management Plan*, Nevada Operations Office, May 1991.

Exhibit 3.0-1. Planning Stage Activities and Work Products by Project Size

Work Activity		Project Size L	Project Size M	Project Size S	Work Product	Scheduled Deliverables L	Scheduled Deliverables M	Scheduled Deliverables S
3.1	Analyze User Environment	R	R	R	Description of user environment Project File	I N	I N	I N
3.2	Define Project Objectives	R	R	R	Statement of project objectives	I	I	I
3.3	Define Project Scope	R	R	R	Statement of project scope	I	I	I
3.4	Develop High-Level Project Requirements	R	R	R	Statement of high-level requirements	I	I	I
3.5	Establish Communication With Functional Areas	R	R	A	Functional area contact list and project profile	N	N	N
3.6	Determine Project Feasibility	R	R	R	Summary of platform options Statement of feasibility Analysis of Benefits and Costs Report Feasibility Study Document	I R A A	I R A A	A A A A
3.7	Develop Project Plan	R	R	A	Project Plan	R	R	A
3.8	Develop Software Quality Assurance Plan	R	R	R	Software Quality Assurance Plan	R	R	R
3.9	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
3.1	Conduct Planning Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
S = Small

Minimum Requirements: R = Required
A = As Appropriate
N = Not Applicable

I = Input to other deliverables
[1] = Completed by reviewer

Activity: **3.1**
Analyze User Environment

Responsibility: Project Manager/Team

Description: A thorough understanding of the current users' environment is necessary to define the objectives, scope, and high-level requirements of the project. Analyze the users' manual procedures or automated processes to understand what users do, how they do it, and what improvements are desired or needed. This includes gaining an understanding of the functions performed, identifying information flows within the processes, and listing process inputs and outputs.

Use appropriate data collection techniques such as user surveys, interviews, and document inspections to gather data and analyze the user environment.

Types of Information: The following list provides samples of the type of information that should be considered.

- Mission - Describe the mission of the primary user organization(s) and how the organizational mission fits into the Departmental mission and strategic plans.

- Work Processes - Analyze the work processes or tasks that are performed by the users. Identify the relationships and priority of the processes.

- Workload - Describe the volume of work currently being performed. For automated processes include processing time for batch operations, response times, peak number of simultaneous users of interactive systems, and number of transactions.

- Processing/Data Flow - Analyze the major processing/data flow for the work processes. Include the flow of data between different user groups, manual and automated processes, and different user sites.

- Integration/Interfaces - Identify interactions and interfaces that the users' current automated systems share with other automated systems.

- Users - Identify the skill levels and number of personnel at both Headquarters and field sites who operate, maintain, and use current manual procedures or automated processes.

Types of Information,

continued:

- Costs - Itemize costs incurred in operating the users' current manual or automated systems.
- Equipment - Identify equipment used in the current manual or automated systems and relate equipment to the function it supports in the systems.
- Software - Identify software packages that are being used.

Work Products:

A substantial amount of information that may be useful in later stages in the software engineering process is gathered during the Planning Stage. Create a centrally maintained Project File that can be used as the repository for all project information gathered during the Planning Stage and for all work products developed throughout the project lifecycle. The Project Manager should verify that all pertinent project information and documentation are placed in the Project File on a timely basis.

Develop a description of the user environment and place a copy in the Project File. The description will be incorporated into future work products such as the Project Plan and the Requirements Specifications.

Activity: **3.2**
Define Project Objectives

Responsibility: Project Manager/Team

Description: Use the information gathered during the analysis of the current user environment to define the objectives of the project. The objectives should identify what the project is intended to accomplish and why it is being undertaken (e.g., to resolve problems or to satisfy statutory requirements). Include a description of any deficiencies in the current manual and automated processes, the severity and impact of any problems, and the solutions and benefits that will result from implementing the project. The objectives should be identified in measurable terms.

Sample Questions: The following list provides sample questions that can be used to help define the project objectives. Even though the users' answers to some questions might be tentative, partial answers will be useful at this stage of the lifecycle. These questions can be revisited during the Requirements Definition Stage to help develop the project requirements.

- What is the general intent of the product?
- What organizational or Departmental functions will the product support?
- What are the major functional components of the product?
- Will the product produce any files or reports or provide data for other Government agencies, organizations, applications?
- Will the product use any data, files, or reports generated by other Government agencies, organizations, applications?
- What Departmental mission(s) will the product support?
- What Departmental strategic goal(s) will the product support?
- Will the product be aligned with the Departmental IRM plans?
- Will the product satisfy statutory or regulatory requirements?
- What are the anticipated benefits of the product?

Work Product: Develop a formal statement of project objectives. This statement will be incorporated into the Project Plan. If a feasibility study is conducted, the statement of project objectives should be included in the Feasibility Study Document. Place a copy of the project objectives in the Project File.

Sample Statement of Project Objectives: *The Human Resources organization needs a reliable means for tracking information about all of the personnel for whom it is responsible. This will be accomplished with the development of a new automated administrative information application called the Human Resources Information System (HRIS). The high-level objectives of the HRIS project are to develop an application that will:*

- *Provide a central local area network repository for personnel data.*
- *Ensure accuracy and timeliness of all personnel data.*
- *Enable access to the data by the Human Resources staff as well as Training, Salary Administration, and other organizations as required.*
- *Generate a series of basic reports.*
- *Provide the capability to produce ad hoc reports in the event existing reports do not satisfy the users' reporting requirements.*

Activity: **3.3**
Define Project Scope

Responsibility: Project Manager/Team

Description: The project scope details what user processes, organizations, and functions will be affected by the product. It also identifies the anticipated changes to current automated and manual processes. A thorough understanding of the scope of the project is necessary to determine whether the project is feasible. The scope may need to be downsized to remain feasible within the constraints of resources, budget, and time negotiated with the system owner.

Sample Questions: The following list provides sample questions that can be used to help determine the project scope. Even though the users' answers to some questions might be tentative, partial answers will be useful at this stage of the lifecycle.

- How many Government/contractor employees will use the product?
- What are the locations of the employees who will use the product?
- What tasks will be performed using the product?
- What will be the operating schedule for the product?
- How many and what types of reports will be needed?
- How and when will reports be distributed? Who receives the reports?
- What query capabilities are needed?
- Are major changes in requirements anticipated in the next few years?
- Are major changes in level of use anticipated in the next few years?
- What is the estimated life expectancy of the product?
- What are the security requirements for the product?
- Will the product be mission-essential to DOE or mission-critical for the system owner?
- Will the product contain vital records?

Sample Questions, continued:

- What are the disaster recovery requirements for the product?
- Will the product require telecommunications?

Work Product: Develop a formal statement of project scope. This statement will be incorporated into the Project Plan. If a feasibility study is conducted, the statement of project scope should be included in the Feasibility Study Document. Place a copy of the project scope in the Project File.

Sample Statement of Project Scope:

Once personnel information is stored on the local area network, the HRIS application can be used as the reliable source of personnel information for all other Human Resources administrative systems.

The use of HRIS by other systems is dependent on the implementation of a client/server environment. This is outside the scope of the HRIS project.

It is outside the scope of HRIS to make electronic feeds to the PERSONA system. The PERSONA and EXMIS extract programs will have to be modified to extract historical information.

Once the HRIS data base is established, EXPATS will need to be redesigned to utilize the HRIS personnel data. Any redesign of EXPATS is outside the scope of the HRIS project.

Users requiring Windows installation or training will be identified; however, the actual installation and training is outside the scope of the HRIS project.

Activity: **3.4**
Develop High-Level Project Requirements

Responsibility: Project Manager/Team

Description: High-level requirements should be of sufficient detail to make a preliminary determination about the feasibility of the project, to estimate the resources that are needed, to assess hardware and software requirements, and to estimate the need for equipment or software training.

The current and anticipated needs of all user groups must be identified. Users in different organizational units or geographic locations may have diverse or unique requirements that must be incorporated into the project requirements.

Sample Requirements: Organize high-level project requirements into categories of related data. The following list provides samples of the types of data that should be considered.

- Inputs - Identify source documents and data that will be used as input to the processes. Provide descriptive information about data such as the type, volume, condition (e.g., edited or unedited), organization, and frequency. Include inputs such as records or batch files from other systems that will be downloaded or migrated.

- Outputs - Identify outputs such as reports, display screens, documents, and data files.

- Data bases - Estimate the high-level contents, purpose, use, format, organization, and update frequency of data bases that will be used by the product. Identify other existing or planned data bases that would interface with the product as a provider or recipient of information.

- Processing/Data Flow - Describe the major processing/data flow for the product. Include flow of data from the product to other systems and vice versa.

- Data Communications - Estimate the major data communications resources required to support the product. Include requirements for networks, dial-up access, and other communication configurations to support data access and retrieval requirements.

Sample

Requirements, continued:

- Interfaces - Identify any systems with which the product must interface. Describe factors that may impact the design of the product.

- Security, Privacy, and Control - State requirements for ensuring the integrity of the data, for safeguarding against unauthorized access to the data bases, and for other user access controls.

- Training - Identify the type of training required to ensure efficient operation of the software product. Provide estimates of the number of personnel to be trained by type and frequency of training.

- Workload - Estimate the volume of work to be handled at slow, normal, and peak periods. Identify dates associated with each period. Include processing time for batch systems, response times, peak number of simultaneous users of interactive systems, and number of transactions.

- Costs - Estimate initial development costs and expected operating cost savings over the expected lifetime of the software product.

- Equipment - Estimate new equipment that might need to be acquired or manufactured and current equipment that would continue to be used.

- Software - Estimate software and firmware packages that might need to be acquired and any updates needed for existing software.

Work Product: Develop a formal statement of the high-level project requirements. This statement will be incorporated into the Project Plan. If a feasibility study is conducted, the statement of requirements should be included in the Feasibility Study Document. The high-level requirements will serve as the foundation for the software requirements developed during the Requirements Definition Stage. Place a copy of the high-level requirements in the Project File.

Reference: The system owner organization's information resource management long-range plan provides useful planning information for consideration when developing the requirements.

Sample Statement of High-Level Requirements: *The following are the high-level access requirements for the HRIS project.*

- *Allow any user to access the application and enter an access request.*
- *Have an interface to verify and maintain user information.*
- *Design system to verify user access levels.*
- *Allow for electronic authorizations for request verification.*
- *Allow for the entry, query, and maintenance of application data based on the user access levels.*
- *Provide for the capture and tracking of request data for the following request types:*
  - *requesting initial computer access*
  - *adding access levels to an existing logon identification code*
  - *reinstating a suspended computer access*
  - *deleting an existing computer access*
  - *suspending an existing computer access*
- *Provide for the entry, query, and maintenance of the following information:*
  - *computer systems*
  - *applications*
  - *user logon identification codes*
- *Allow users to view and maintain their own address information*
- *Provide a means for the system owner and security officers to review and change current user access information*

Activity: **3.5**
Establish Communication With Functional Areas

Responsibility: Project Manager

Description: Early contact with the functional areas that will provide input to, or support for, the project is necessary for developing accurate estimates of the project scope, cost, resources, and schedule. Representatives of these functional areas should be involved in all stages of the project lifecycle and are participants in the Stage Exit process.

Develop a brief profile about the software project. Provide enough information so that the points-of-contact in each functional area will be able to estimate support requirements and resource allocations for the project. A sample project profile form is provided at the end of this section.

Develop a list of all functional areas and points-of-contact who will provide input to, or support the project. Send each point-of-contact the project profile and request input from all recipients.

Note: This activity is not necessary for small software engineering projects that do not require input from other functional areas.

Work Products: Place a copy of the project profile in the Project File. Update the project profile as needed to maintain an accurate description of the software product.

Keep the list of functional area contacts current and maintain a copy in the Project File. Use this list as the starting point when functional areas need to be contacted about involvement in project activities such as Stage Exits.

Software Project Profile

Software Name/Identification: _____ Acronym: _____

Contract Number: _____ Task Number: _____

Project Manager: _____ Telephone: _____ E-mail Address: _____
DOE Organization: _____ POC: _____ Telephone: _____

Size of effort *(as defined by Software Engineering Methodology)*: ☐ Large ☐ Medium ☐ Small

Purpose and scope of software product: _____

_____
_____

_____
_____

1. Software processing platform(s) *(check all that apply)*

   ☐ Mainframe ☐ Minicomputer ☐ LAN Server: _____
   ☐ Microcomputer Type: ☐ IBM compatible ☐ Macintosh ☐ Other: *(specify)* _____

2. The software will run under the following operating environments *(check all that apply)*

   ☐ VM ☐ DOS ☐ Windows ☐ Novell
   ☐ MVS ☐ OS/2 ☐ Windows N/T ☐ Unix
   ☐ CICS ☐ Macintosh ☐ Windows 95 ☐ Sun

   ☐ Other *(specify)* _____

3. The software will support the following printers *(check all that apply)*

   ☐ Workstation printer ☐ LAN-shared ☐ Mainframe/minicomputer addressable

4. The following programming languages(s), data base management system, or file system are being considered.

   Programming Language: _____
   Data Base Management System: _____
   File System: _____

(more)

5. The software will interface with the following software or data sources (sharing, receiving, or sending data).

_____
_____
_____
_____

6. The following mechanisms are planned for providing these interfaces (e.g., Internet, TCP/IP, LAN).

_____
_____
_____

7. The users of the software are at the following locations *(check all that apply)*

☐ DOE Headquarters ☐ DOE Operations Office ☐ National Lab

☐ Other (specify): _____

8. Software/Data Sensitivity: ☐ Classified ☐ Unclassified Sensitive ☐ Unclassified

9. The data is critical to the mission of the organization.

☐ Yes ☐ No

10. The data is essential to the mission of DOE and must be available at all times.

☐ Yes ☐ No

11. The following training support services are anticipated to develop and install the software product.

☐ Consultation ☐ Classroom ☐ Self-Study ☐ Computer-Based

12. The following documentation support services are anticipated to complete the project.

☐ Writing ☐ Editing ☐ Graphics ☐ Production

Additional comments:

_____
_____
_____

Activity: **3.6**
Determine Project Feasibility

Responsibility: Project Manager/Team

Description: In this stage, the feasibility of successfully developing and implementing the project is determined. Project feasibility leads to a "go" or "no go" decision about the project. Determining project feasibility is an interactive process of collecting and analyzing data and searching for cost-effective, viable technical solutions.

Use the project objectives, scope, and high-level requirements as the basis for determining project feasibility. Work with the user organization and functional area representatives to address technical issues and risks. Conduct research and investigate documents and other resources.

Note: Feasibility may not be an issue for some small software development projects. A Feasibility Review is not required when feasibility is obvious.

Sample Questions: The following is a list of sample questions that can be used to help determine the feasibility of a project.

- Can the users needs/problems best be satisfied with a manual process, automated process, or combination?
- Is it cost-effective to develop an automated process?
- Is the scope of the project feasible within time, resource, and hardware and software constraints and limitations?
- Is there at least one technically feasible automated solution for the project?
    - If a project is well defined and has no automation issues, a single straightforward automated solution may sufficiently demonstrate cost and technical feasibility.
    - Where automation issues have been identified, technical alternatives should be associated with each proposed solution.

Tasks: The following tasks are involved in determining project feasibility.

3.6.1 Investigate Software Alternatives
3.6.2 Investigate Hardware Alternatives
3.6.3 Formulate Platform Options
3.6.4 Conduct Feasibility Review
3.6.5 Conduct Analysis of Benefits and Costs
3.6.6 Conduct Feasibility Study

Task: **3.6.1**
Investigate Software Alternatives

Description: When the software to be used for the project has not been predetermined by the system owner's existing computing environment, software available within the Department and the commercial marketplace should be investigated. In the Planning Stage, the investigation of software alternatives is geared to determining project feasibility.

Unless the cost effectiveness of developing custom-built software to meet mission needs is clear and documented, all sources of reusable code, applications, and commercial off-the-shelf software must be investigated on a site and Departmentwide basis prior to making a decision to custom-build code for the project. This practice ensures the most cost-effective and efficient use of resources, and will decrease the number of duplicative and overlapping software systems. The choice to develop a customized application should be balanced against the availability of other solutions; and the project cost, resources, and time constraints.

Software Alternatives: Information on software products or modules can be obtained by notifying field sites, DOE Headquarters, other Government agencies, and private industry via Internet. The following is a list of software alternatives that should be considered.

- Adapt existing software in use within the Department.
- Adapt existing software in use within other Government agencies.
- Adapt mainframe or minicomputer source code obtained from Departmental Computer Center repositories.
- Purchase commercial off-the-shelf software.
- Reuse existing modules of code.
- Adapt reusable code to fit the new application.
- Develop a custom-built software product.

Exhibit 3.6-1, Checklist for Investigating Software Alternatives, provides a checklist for investigating existing software resources.

Note: Medium and small software engineering efforts are often restricted to the system owner's existing software. This should not preclude the potential cost savings of reengineering existing software modules rather than custom building the entire software system.

Exhibit 3.6-1. Checklist for Investigating Software Alternatives

Software Resource	Contact	Special Considerations
Review Headquarters software repositories including: -Microcomputer and local area network applications in the Microcomputer Application Systems Library (MASL) -Applications in the Systems Review Inventory System (SRIS)	-MASL Coordinator -Administrative Computer Center SRIS listing on HR-01 LAN	If the requirements are met by reusable software obtained from MASL, SRIS, or SASREPS, then adapting that software may provide the best solution. If the source application is very similar to the target application, was developed to standards, and includes documentation, this can be a very attractive alternative to a custom-built application. Reusable code may be customized to meet specific requirements.
Review field site software repositories	Field site repository owner/coordinator	
Review the Department of Commerce Interagency Survey of Administrative Systems Reporting System (SASREPS)	MASL Coordinator at DOE Headquarters	
Contact other project teams working on similar projects	Project managers	

Exhibit 3.6-1. Checklist for Investigating Software Alternatives

Software Resource	Contact	Special Considerations
Investigate off-the-shelf commercial software	Software vendors	Request and evaluate vendor-supplied demonstrations of software products that may satisfy project requirements. Commercial software may offer a macro language or program code that can be modified by the developer, which can significantly decrease development time. Some disadvantages to commercial software include procurement lead time, developer and user learning curves, and interfaces with existing hardware and software.
Determine software libraries available for particular software languages	Mainframe software administrators Microcomputer software development groups	Reusable code can be either modules of code that are used as written, or units of code that are reengineered to perform a similar function. If the reusable tools for the particular development language are numerous, this alternative may be more attractive than modifying an application that will need considerable customization.
Reuse modules of code or software libraries developed within your functional area	Task leaders and other programmers	Some software engineering teams develop modules of reusable code that are language specific and perform functions such as screen formats, data validation, error-handling, data access, and other frequently used routines. Once developed, tested, and refined, these routines become reliable building blocks for the rapid development of other applications.

Task: **3.6.2**
Investigate Hardware Alternatives

Description: When the hardware to be used for the project has not been predetermined by the system owner's existing computing environment, investigate hardware available within the Department and through the commercial marketplace. In the Planning Stage, the investigation of hardware is geared to determining project feasibility.

Factors to Consider: The following is a list of factors that should be considered when identifying hardware alternatives.

- Availability and cost of hardware
  - Shareable hardware
  - Government excess
  - New procurement
- Current and future telecommunications needs
- Computer security requirements of the system
- Volume of data
- Importance of data to the Departmental mission
- Importance of data to the user organization's mission and to job performance
- Potential growth of the software to serve more users
- Potential growth of the software to serve more locations
- Potential for interface to other systems or organizations
- Conformance to Government standards such as networking and open systems

Note: Medium and small software engineering efforts are often restricted to the system owner's or user sites' existing hardware.

Task: **3.6.3**
Formulate Platform Options

Description: Use the information collected about software and hardware alternatives to formulate preliminary platform options. The purpose of identifying platform options early in the project lifecycle is to assure that at least one technically feasible and cost-effective approach exists to satisfy the project objectives. If more than one platform option is feasible, identify the benefits, costs, assumptions, constraints, dependencies, and risks associated with each option.

No platform decisions are made at this time. Detailed technical solutions are premature prior to defining the product requirements. The platform alternatives information gathered in the Planning Stage is revisited in the Functional Design Stage, at which time a final recommendation is developed by the project team and presented to the system owner. The system owner is responsible for making the final platform decision.

Work Product: Develop a summary of platform options for use in the Feasibility Review or Feasibility Study. Place a copy of the platform option information in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the most viable platform options have been identified.

Task: **3.6.4**
Conduct Feasibility Review

Description: A Feasibility Review is an informal meeting to determine whether the software project can be accomplished with the available resources, system owner and users' computing environment, and technological constraints. The Feasibility Review meeting also provides an opportunity for project management to obtain feedback from other project managers and the functional area representatives who will be providing input to, or supporting, the project throughout the lifecycle.

The project objectives, scope, high-level requirements, and preliminary platform options should be shared with the review meeting participants prior to the meeting date. The participants are expected to evaluate the project information and risks, and make a recommendation about project feasibility.

Feasibility Factors: The following are some typical factors that should be considered when determining the feasibility of a project.

- Project scope and objectives
- Users' computing environment
- High-level requirements
- Assumptions, constraints, and limitations
- Platform options
- Security and recovery objectives
- Risk factors
- Technological factors
- Available resources and budget
- Future growth needs
- Expected long-term benefits
- Compliance with long-range information resource management plans

Recommendations: After all of the pertinent feasibility factors have been considered, the review meeting participants should make one of the following recommendations:

- Proceed with the project without performing a Feasibility Study
- Prepare an Analysis of Benefits and Costs
- Conduct a Feasibility Study, which includes an Analysis of Benefits and Costs
- Stop the project

Work Products: Generate a record of the Feasibility Review meeting to serve as verification that the review occurred, to record feasibility factors that were considered and the recommendation(s) generated during the meeting, and to provide background information if a Feasibility Study or Analysis of Benefits and Costs is required.

The project manager uses the recommendations from the Feasibility Review meeting to develop a formal statement of feasibility. A typical statement of feasibility is a short declaration describing whether or not it is feasible to develop the project within the known constraints. When major risks are involved in the feasibility decision, it may be necessary to expand the statement of feasibility to describe the risk factors and their consequences.

Depending on the factors that must be considered for each project, the statement of feasibility may contain the following information.

- Project objectives
- Summary of issues concerning:
  - development and implementation
  - assumptions, constraints, and limitations
  - project scope
- Results of research on hardware and software alternatives
- Significant risk factors
- Feasibility recommendation(s)

The project manager decides on the final recommendation and reports the findings to the system owner for review and approval.

Sample Feasibility Statement: The following is a sample feasibility statement for a low-risk project that would use the hardware/software platform currently available within the users' organization.

The client organization, Project Management Officer, and project manager agree that the XYZ project will be written in {programming language} and use {operating system and/or DBMS} on {hardware configuration}, all of which are currently in place and can easily absorb the impact of XYZ. This will be a custom-built product since a search of software repositories did not reveal any reusable or existing software that would satisfy the project requirements.

Task: **3.6.5**
Conduct Analysis of Benefits and Costs *(as appropriate)*

Description: An Analysis of Benefits and Costs (ABC) is a useful tool in any stage of the software lifecycle. In the Planning Stage, the results of an ABC help to determine the feasibility of a project and the return on investment. For example, an ABC can be conducted to determine if changing the users' current business processes or computing environment will improve efficiency or reduce overhead expenditures enough to justify the cost of the project, and when the system owner can expect to recoup the costs of the project in benefits.

An ABC is used to identify and compare the benefits and costs associated with all of the hardware or software alternatives. Any advantage to a particular alternative is considered a benefit, and any loss or penalty is considered a cost. Costs can also include the purchase price of supplies, equipment, software, personnel time or charge rate, and system downtime. The results of the ABC indicate the most cost-effective alternative.

When a totally manual process is being automated, the benefits of automating the process may be obvious. If the system owner has restricted the platform, then an ABC can be an appropriate way to document these decisions and the benefits and costs associated with the limitations.

When a Feasibility Study is performed, an ABC is a mandatory requirement of the study. When a Feasibility Study is not performed, the ABC is an optional process.

Work Product: Develop a report that describes the results of the ABC. When a Feasibility Study is performed, the results of the ABC will be incorporated into the Feasibility Study Document.

Review Process: An informal peer review or a structured walkthrough is recommended to validate the ABC process used and the results obtained.

Reference: The following Department documents provide guidance on conducting an Analysis of Benefits and Costs.

- Analysis of Benefits and Costs (ABC's) Guideline: Volume 1, A Manager's Guide to Analysis of Benefits and Costs
- Analysis of Benefits and Costs (ABC's) Guideline: Volume 2, An Analyst's Handbook for Analysis of Benefits and Costs

Task: **3.6.6**
Conduct Feasibility Study *(as appropriate)*

Description: When a project has decisions or issues that require a more detailed investigation than is possible with a Feasibility Review or Analysis of Benefits and Costs, a Feasibility Study must be performed to obtain the necessary information for making an informed decision about project feasibility. An Analysis of Benefits and Costs (ABC) is a required process in a Feasibility Study.

In cases where the platform is limited or restricted, the Feasibility Study may be abbreviated to evaluate only the technical solutions for the areas that have some flexibility.

Use the information identified in the Feasibility Review and the Analysis of Benefits and Costs as the basis for the Feasibility Study. Consider any preliminary solutions that were formulated and identify the alternative ways to resolve the problems or issues. Evaluate all of the available feasibility factors to determine if the project is technically feasible and cost effective.

Sometimes a Feasibility Study for a similar project has already been conducted. An existing Feasibility Study can be used if the information is current, relevant to the new project, and technically correct.

The following are examples of cases where a Feasibility Study must be performed.

- There is uncertainty or disagreement on the boundaries of the project.
- There is uncertainty over the cost justification or technical feasibility of a project.
- There is a lack of agreement about the goals or approach for building the software product.
- The proposed size or complexity of the software product indicates a high degree of risk.
- The software product will automate functions that currently are not being performed either automatically or manually.

Work Product: The results of the Feasibility Study are reported in a document that describes the process that was used to determine feasibility, the alternatives that were considered, and the results of the Analysis of Benefits and Costs. The Feasibility Study results determine the feasibility recommendation for the project.

Subtasks: The following subtasks are involved in conducting and documenting a Feasibility Study.

3.6.6.1 Analyze the alternatives
3.6.6.2 Determine feasibility recommendation
3.6.6.3 Develop feasibility study document

Note: New software products can be limited to the system owner's and users' existing hardware and software environment, and may not require a Feasibility Study.

Review Process: An informal peer review or a structured walkthrough is recommended to validate the Feasibility Study process used and the results obtained.

Subtask: **3.6.6.1**
Analyze the alternatives

Description: Alternatives for developing and implementing a project are derived from the high-level project requirements, the results of the Feasibility Review and the Analysis of Benefits and Costs, and the preliminary platform options. The analysis of the alternatives forms the basis for determining project feasibility.

The analysis of the alternatives should consider the following types of information.

- The ability of each alternative to achieve the project objectives.
- The ability of each alternative to meet the users' requirements and expectations.
- How well each alternative accommodates the system owner's current processes and resources.
- How cost-effective and technically feasible each alternative is compared to the existing automated or manual process.
- How well each alternative fits with the hardware and software limitations imposed by the system owner.

Analysis of the alternatives may include the following activities.

- Research current computer industry periodicals to obtain articles and reviews about software and hardware alternatives.
- Interview software and hardware vendors to obtain up-to-date information about product releases and future upgrades, capabilities, vendor support, developer training, product demonstrations, multiuser license arrangements, current users, and costs.
- Interview current users of the product to obtain information about user satisfaction, ease-of-use, satisfaction of user expectations, productivity, and product limitations.

Subtask: **3.6.6.2**
Determine feasibility recommendations

Description: The results of the Feasibility Study are used to determine project feasibility. The feasibility recommendations must be substantiated by the results of the Analysis of Benefits and Costs (ABC).

The feasibility recommendations should include the following types of information.

- The recommended alternative for each of the project automation issues.
- The feasibility to develop the project.
- The most technically sound alternative with the most long-range benefits to the Department.
- The most cost-effective configuration for the project based on the ABC.
- The estimated total lifecycle costs based on the recommended technical solution and the ABC.

Subtask: **3.6.6.3**
Develop feasibility study document

Description: The Feasibility Study Document provides the following types of information.

- The process that was used to determine project feasibility.
- The alternative approaches that were analyzed for achieving the project objectives.
- The results of the Analysis of Benefits and Costs.
- The recommendations for a specific approach to meet the system owners' and users' project objectives, automation needs, and expectations.

Work Product: The Feasibility Study Document should contain enough information to enable the system owner to make a decision to either continue or terminate the project.

Review Process: An informal peer review or a structured walkthrough is recommended to validate the Feasibility Study Document and feasibility recommendations.

The completion of the Feasibility Study is an appropriate time to schedule an In-Stage Assessment (ISA). The *In-Stage Assessment Process Guide* provides a description and instructions for conducting an ISA. A copy of the guide is provided in Appendix D.

Activity: **3.7**
Develop Project Plan

Responsibility: Project Manager

Description: The purpose of the Project Plan is to establish reasonable plans for performing the software engineering activities and for managing and tracking the software project. The following project management activities must be performed before a Project Plan document is developed.

- Define the management approach for the project including project tracking and oversight activities
- Formulate the technical approach for the project
- Develop the project estimates
- Establish the project development team

The results of these project management activities and the lifecycle activities described in this volume will provide input for the Project Plan.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Work Product: Develop a Project Plan that provides detail for the Planning and Requirements Definition Stages and high-level information for the other lifecycle stages. At the conclusion of each stage, the Project Plan will be reviewed to determine if the project estimates for resources, cost, and schedule need to be revised for either the current stage or subsequent stages. In addition, the Project Plan will be expanded to provide detailed estimates of resources, costs, and hours for the next stage. A Project Plan Example document template is accessible online to use as a guide in developing the Project Plan. Project models containing lifecycle stages and deliverables that can be used as is or tailored for any software development or maintenance project are accessible online. The files can be downloaded for use with Microsoft Project.

Review Process: Conduct a structured walkthrough to ensure that the Project Plan reflects the project objectives and scope; identifies and mitigates project risks, and adequately estimates the project resources, costs, and schedule.

Activity: **3.8**
Develop Software Quality Assurance Plan

Responsibility: Project Manager and Quality Assurance Manager

Description: The purpose of quality assurance is to assure the production and operation of high quality products on schedule, within budget, and within the constraints specified by the system owner and user. The software quality assurance program is initiated at the beginning of a project and is conducted throughout the software engineering lifecycle. The software quality assurance program is the joint responsibility of the project manager and quality assurance manager with direct support and involvement from the quality assurance practitioners assigned to the project.

Work Product: The quality assurance manager or designated representative assists the project manager with the development of a plan that clearly defines the project's quality assurance policies and procedures. The Software Quality Assurance Plan addresses the following types of responsibilities.

- Establishing the applicability of published standards and procedures and determining the scope of the project standards and procedures.
- Monitoring the software product and enforcement of compliance with all standards and procedures to facilitate the early detection of problems that could affect the reliability, maintainability, availability, integrity, safety, security, or usability of the software product.
- Inspecting hardware and software items and documenting for compliance to specifications and standards before their release to the test team or the system owner.
- Certifying deliverable items before their release to the system owner as compliant with all provisions of the project statement of work and contract, if applicable.
- Coordinating the project's technical problem reporting system and corrective action program to assure resolution of observed discrepancies.
- Measuring the quantitative and auditable progress of the project based on cost, schedule status, and quality status.

Work Product, continued:

- Assuring consistent management and technical practices and the integrity of the software product.

Provide enough information in the plan so that compliance can be monitored by means of project records. Whenever feasible, acquire automated tools to check compliance with project standards. For example, many CASE (computer-aided software engineering) tools can check compliance with standards, while checking the validity and consistency of requirements, design, and logic diagrams.

Review Process: Conduct a structured walkthrough to validate that the quality assurance policies and procedures are appropriate and adequate for the project.

Activity: **3.9**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the Planning Stage Exit process. Additional ISAs can be performed during the stage, as appropriate. An ISA is recommended after the completion of the Feasibility Study.

Provide the reviewer with copies of all work products developed or revised during the Planning Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/best practices.
- Sound project management practices are being used.
- The project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide.* A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **3.10**
Conduct Planning Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the Planning Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the guide is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the Requirements Definition Stage.

Software Engineering Methodology

Chapter 4.0
Requirements Definition Stage

Table of Contents

Chapter **Page**

4.0 Requirements Definition Stage 4.0-1
- 4.1 Develop Software Configuration Management Plan 4.1-1
- 4.2 Select Requirements Analysis Methodology 4.2-1
- 4.3 Define Project Requirements 4.3-1
  - 4.3.1 Define Functional Requirements 4.3-3
  - 4.3.2 Define Input and Output Requirements 4.3-5
  - 4.3.3 Define Performance Requirements 4.3-6
  - 4.3.4 Define User Interface Requirements 4.3-7
  - 4.3.5 Define System Interface Requirements 4.3-8
  - 4.3.6 Define Communication Requirements 4.3-9
  - 4.3.7 Define Computer Security and Access Requirements 4.3-10
  - 4.3.8 Define Backup and Recovery Requirements 4.3-13
  - 4.3.9 Define Data Requirements 4.3-16
  - 4.3.10 Define Implementation Requirements 4.3-17
- 4.4 Compile Project Requirements 4.4-1
  - 4.4.1 Develop Requirements Traceability Matrix 4.4-2
  - 4.4.2 Develop Software Requirements Specification 4.4-4
- 4.5 Establish Functional Baseline 4.5-1
- 4.6 Develop Project Test Plan 4.6-1
  - 4.6.1 Identify Test Methodologies 4.6-3
  - 4.6.2 Identify Test Phases 4.6-4
  - 4.6.3 Identify Test Environment Requirements 4.6-6
- 4.7 Develop Acceptance Test Plan 4.7-1
- 4.8 Select Design Methodology 4.8-1
- 4.9 Revise Project Plan 4.9-1
- 4.10 Conduct In-Stage Assessment 4.10-1
- 4.11 Conduct Requirements Definition Stage Exit 4.11-1

Chapter: **4.0**
Requirements Definition Stage

Description: The primary goal of this stage is to develop a basis of mutual understanding between the system owner/users and the project team about the requirements for the project. The result of this understanding is an approved Software Requirements Specification that becomes the initial baseline for software product design and a reference for determining whether the completed software product performs as the system owner requested and expected.

This stage involves development of a Software Configuration Management Plan to track and control work products, analysis of the system owner/users' business processes and needs, translation of those processes and needs into formal requirements, and planning the testing activities to validate the performance of the software product.

Input: The following work products provide input to this stage.

- Project File
- Description of user environment
- Statement of project scope and objectives
- Statement of high-level project requirements
- Functional area contact list and project profile
- Summary of platform options
- Statement of project feasibility
- Analysis of Benefits and Costs Report
- Feasibility Study Document
- Project Plan
- Software Quality Assurance Plan

High-Level Activities: The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage. These activities represent the minimum requirements for a large software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

4.1 Develop Software Configuration Management Plan
4.2 Select Requirements Analysis Methodology
4.3 Define Project Requirements
4.4 Compile Project Requirements

High-Level Activities, continued:

4.5 Establish Functional Baseline
4.6 Develop Project Test Plan
4.7 Develop Acceptance Test Plan
4.8 Select Design Methodology
4.9 Revise Project Plan
4.10 Conduct In-Stage Assessment
4.11 Conduct Requirements Definition Stage Exit

Output: Several work products are developed during this stage. The work products listed below are the minimum requirements for a large software project. Deviations in the content and delivery of these work products are determined by the size and complexity of a project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

- Software Configuration Management Plan
- Description of analysis methodology
- Records of all project requirements
- User-oriented requirements manual *(optional)*
- Continuity of Operations Statement/Plan
- Data Dictionary
- Requirements Traceability Matrix
- Software Requirements Specification
- Project Test Plan
- Acceptance Test Plan *(draft)*
- Design methodology
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 4.0-1, Requirements Definition Stage Activities and Work Products by Project Size.* The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large products.

Review Process: Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. The time and resources needed to conduct the walkthroughs should be reflected in the project resources, schedule, and work breakdown structure.

Exhibit 4.0-1. Requirements Definition Stage Activities and Work Products by Project Size

Work Activity		Project Size L	Project Size M	Project Size S	Work Product	Scheduled Deliverables L	Scheduled Deliverables M	Scheduled Deliverables S
4.1	Develop Software Configuration Management Plan	R	R	R[2]	Software Configuration Management Plan	R	R	R[2]
4.2	Select Requirements Analysis Methodology	R	R	R	Description of analysis methodology	I	I	I
4.3	Define Project Requirements	R	R	R	Records of project requirements User-oriented requirements manual *(optional)* Continuity of Operations Statement/Plan Data Dictionary	I O R R	I O R R	I O R R
4.4	Compile Project Requirements	R	R	R	Requirements Traceability Matrix Software Requirements Specification *(draft)*	R R	R R	A R
4.5	Establish Functional Baseline	R	R	R	Software Requirements Specification *(final)*	R	R	R
4.6	Develop Project Test Plan	R	R	A	Project Test Plan	R	R	A
4.7	Develop Acceptance Test Plan	R	R	R	Acceptance Test Plan *(draft)*	R	R	R
4.8	Select Design Methodology	R[2]	R[2]	R[2]	Design Methodology	R[2]	R[2]	R[2]
4.9	Revise Project Plan	R	R	A	Project Plan *(revised)*	R	R	A
4.1	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
4.11	Conduct Requirements Definition Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
S = Small

Minimum Requirements: R = Required
A = As Appropriate
N = Not Applicable

I = Input to other deliverables
O = Optional work product
[1] = Completed by reviewer
[2] = Can use existing plan

Reference: *Appendix C, Conducting Structured Walkthroughs,* provides a procedure and sample forms that can be used for structured walkthroughs.

Bibliography: The following materials were used in the preparation of the Requirements Definition Stage chapter.

1. Bramucci, Wilma, "Systems Development Software," *Faulkner Technical Reports, Inc.*, June 1989. pp. 1-7.

2. Fairley, Richard E., *Software Engineering Concepts,* McGraw-Hill Book Company, New York.

3. *Software Engineering Handbook,* Chapter 4, Software Requirements Analysis.

4. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications,* ANSI/IEEE Std 830-1984, New York, 1984.

5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.

6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans,* ANSI/IEEE Std 1012-1986, New York, 1986.

7. U.S. Department of Commerce, National Bureau of Standards, *Guideline for Planning and Management of Database Applications,* Federal Information Processing Standards Publication 77, 1980. pp. 10-23.

8. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual,* 1988.

9. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual,* 1988.

Activity: **4.1**
Develop Software Configuration Management Plan

Responsibility: Project Manager

Description: Configuration management is a set of procedures used to control changes to the project during all stages of the software lifecycle. A Software Configuration Management Plan that describes the configuration management procedures is required for each software project. Based on the complexity of the project and the anticipated volume of changes, a plan can be developed for a specific project, an existing plan can be modified to suit the requirements of a project, or a plan can be developed to manage all of the projects supporting a particular system owner's organization.

The plan is developed early in the lifecycle to ensure the control of changes as soon as the project requirements are approved and baselined. In this stage, the plan addresses activities that are platform independent, such as identifying the items that will be placed under configuration management. As the project progresses through the lifecycle stages, the plan is expanded to reflect platform specific activities.

Work Product: Prepare a Software Configuration Management Plan document if an existing plan is not available or applicable for the project. Place a copy of the Software Configuration Management Plan in the Project File.

Review Process: Conduct a structured walkthrough to validate that the configuration management approach, the configuration identification, change control, status accounting, and auditing procedures are appropriate for the project.

Activity: **4.2**
Select Requirements Analysis Methodology

Responsibility: Project Manager/Team

Description: A requirements analysis methodology is the set of data collection and analysis techniques (e.g., user interviews and rapid prototyping) combined with the lifecycle requirements standards (e.g., tracing the requirements through all lifecycle activities) that are used to identify the project requirements and to define exactly what the software product must do to meet the system owner/users' needs and expectations. When appropriate, the methodology must include techniques for collecting data about users at more than one geographic location and with different levels and types of needs.

The requirements analysis methodology should be in harmony with the type, size, and scope of the project; the number, location, and technical expertise of the users; and the anticipated level of involvement of the users in the data collection and analysis processes. The methodology should ensure that the functionality, performance expectations, and constraints of the project are accurately identified from the system owner/users' perspective. The methodology should facilitate the analysis of requirements for their potential impact on existing operations and business practices, future maintenance activities, and the ability to support the system owner's long-range information resource management plans.

It is advantageous to select a methodology that can be repeated for similar projects. This allows the project team and the system owner/users to become familiar and comfortable with the methodology. Discuss the analysis methodology with the system owner and users to make sure they understand the process being used, their role and responsibilities in the process, and the expected format of the output (e.g., how the requirements will be organized and described).

Work Product: Create a description of the analysis methodology and share it with all members of the project team, system owner, and users. Place a copy of the analysis methodology description in the Project File.

Review Process: Conduct a structured walkthrough to verify that the requirements analysis methodology is appropriate for the scope and objectives of the project. A structured walkthrough is not needed when the methodology has been used successfully on similar projects for the same system owner/user environment.

Activity: **4.3**
Define Project Requirements

Responsibility: Project Manager/Team

Description: Use the project scope, objectives, and high-level requirements as the basis for defining the project requirements. The questions used to define the project objectives may be helpful in developing the project requirements. The goals for defining project requirements are to identify what functions are to be performed on what data, to produce what results, at what location, and for whom. The requirements must focus on the software products that are needed and the functions that are to be performed. Avoid incorporating design issues and specifications in the requirements.

Requirements should be specified as completely and thoroughly as possible. The requirements must support the system owner's business needs, information resource management long-range plans, and the organizational and Departmental missions. When requirements are being defined, it is not sufficient to state only the requirements for the problems that will be solved; all of the requirements for the project must be captured.

Attributes: Each requirement must be stated as a unique objective with the following attributes. The existence of these attributes must be verified prior to the delivery of the Software Requirements Specification later in the Requirements Definition Stage.

- Necessary - Absolute requirements that are to be verified are indicated by "must" or "shall". Goals or intended functionality are indicated by "will".
- Correct - Each requirement is an accurate description of a feature or process of the software product.
- Unambiguous - The statement of each requirement denotes only one interpretation.
- Complete - Each requirement describes one result that must be achieved by the software product. The requirement should not describe the means of obtaining the result.
- Consistent - Individual requirements are not in conflict with other requirements.

Attributes, continued:

- Verifiable (testable) - Each requirement is stated in concrete terms and measurable quantities. A process should exist to validate that the software product (when developed) will satisfy the set of requirements.

- Modifiable - The structure and style of the requirements are such that any necessary changes to the requirements can be made easily, completely, and consistently.

- Traceable - The origin of each requirement is clear and can be tracked in future development activities and tests.

Identification System: The creation of a standard identification system for all requirements is required in order to facilitate configuration control, requirements traceability, and testing activities. The identification system must provide a unique designator for each requirement. For example, the identification system can classify the requirements by type (e.g., functional, input, or computer security). Within each type classification, the requirements can be assigned a sequential number. Select an identification system that is appropriate for the scope of the project.

Changes: As the project evolves, the requirements may change or expand to reflect modifications in the users' business plans, design considerations and constraints, advances in technology, and increased insight into user business processes. A formal change control process must be used to identify, control, track, and report proposed and approved changes. Approved changes in the requirements must be incorporated into the Software Requirements Specification in such a way as to provide an accurate and complete audit trail of the changes. This change control process should be an integral part of the project's Software Configuration Management Plan.

Tasks: The following tasks are involved in developing project requirements.

4.3.1	Define Functional Requirements
4.3.2	Define Input and Output Requirements
4.3.3	Define Performance Requirements
4.3.4	Define User Interface Requirements
4.3.5	Define System Interface Requirements
4.3.6	Define Communication Requirements
4.3.7	Define Computer Security and Access Requirements
4.3.8	Define Backup and Recovery Requirements
4.3.9	Define Data Requirements
4.3.10	Define Implementation Requirements

Task: **4.3.1**
Define Functional Requirements

Description: Functional requirements define what the software product must do to support the system owner's business functions and objectives. The functional requirements should answer the following questions.

- How are inputs transformed into outputs?
- Who initiates and receives specific information?
- What information must be available for each function to be performed?

Identify requirements for all functions whether they are to be automated or manual. Describe the automated and manual inputs, processing, outputs, and conditions for all functions. Include a description of the standard data tables and data or records that will be shared with other applications. Identify the forms, reports, source documents, and inputs/outputs that the software product will process or produce to help define the functional requirements.

A functional model should be developed to depict each process that needs to be included. The goal of the functional model is to represent a complete top-down picture of the software product.

Flow diagrams should be used to provide a hierarchical and sequential view of the system owner's business functions and the flow of information through the processes.

Work Product: Maintain a record of all functional requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the functional requirements in the Project File.

Sample Functional Requirement: *The selection criteria for the extraction of records shall be the occurrence of the letters "EW" in the Budget and Reporting Code field.*

Optional Work Product: Consider developing an optional work product that defines how the final software product will operate to support the system owner organization's business functions and objectives. This user-oriented requirements manual would identify processes in a narrative form from the user's perspective and would include requirements for all functions whether they are to be automated or manual. A functional description can be developed to depict each process

Optional Work Product, continued: that will be provided. The goal is to present a complete top-down picture of the software product. This user-oriented requirements manual can be used as an aid in validating the functional requirements and serves as the basis for the user documentation. If a test group outside the project team is used, the test group can work with the project team to develop the manual.

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the functional requirements.

Task: **4.3.2**
Define Input and Output Requirements

Description: Describe all manual and automated input requirements for the software product such as data entry from source documents and data extracts from other applications.

Describe all output requirements for the software product such as printed reports, display screens, and files.

Work Product: Maintain a record of all input and output requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the input and output requirements in the Project File.

Sample Input Requirement: *The application must automatically assign a unique, sequential Employee Number to each employee record that is entered into the data base.*

Sample Output Requirement: *All reports that contain Privacy Act data must include a warning statement in the report header information.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the input and output requirements.

Task: **4.3.3**
Define Performance Requirements

Description: Performance requirements define how the software product must function (e.g., hours of operation, response times, and throughput under detailed load conditions). The information gathered in defining the project objectives can translate into very specific performance requirements; (e.g., if work performed for an organization is mission essential to the Department, the hours of operation and throughput will be critical to meeting the mission). Also, Government and DOE policy can dictate specific availability and response times.

Work Product: Maintain a record of all performance requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the performance requirements in the Project File.

Sample Performance Requirement: *The application must be available for use from 8:00 a.m. to 5:00 p.m. Monday through Friday.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the performance requirements.

Task: **4.3.4**
Define User Interface Requirements

Description: The user interface requirements should describe how the user will access and interact with the software product, and how information will flow between the user and the software product.

Interface Issues: The following are some of the issues that should be considered when trying to identify user interface requirements.

- The users' requirements for screen elements, navigation, and help information.
- The standards for the programmatic organization, DOE, Government, and industry that apply to user interfaces.
- The range of skill levels of the users who will access and use the software product.
- The range of work that the users will be performing with the software product.

Define the user interface requirements by identifying and understanding what is most important to the user, not what is most convenient for the project team. Work with the system owner and users to develop a set of user interface requirements that can be used for all automated products for the system owner's organization. A standard set of user interface requirements will simplify the design and code processes, and ensure that all automated products have a similar look and feel to the users. When other constraints (such as a required interface with another application) do not permit the use of existing user interface standards, an attempt should be made to keep the user interface requirements as close as possible to the existing standard.

Work Product: Maintain a record of all user interface requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the user interface requirements in the Project File.

Sample User Interface Requirement: *All data entry screens must include a unique screen identification number.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the user interface requirements.

Task: **4.3.5**
Define System Interface Requirements

Description: The hardware and software interface requirements must specify hardware and software interfaces required to support the development, operation, and maintenance of the software product. The following information should be considered when defining the hardware and software interface requirements.

- System owner's and users' computing environment.
- Existing or planned software that will provide data to or accept data from the software product.
- Other organizations or users having access to the software product.
- Purpose or mission of interfacing software.
- Common users, data elements, reports, and sources for forms/events/outputs.
- Timing considerations that will influence sharing of data, direction of data exchange, and security constraints.
- Development constraints such as the operating system, data base management system, language compiler, tools, utilities, and network protocol drivers.
- Standardized system architecture defined by hardware and software configurations for organizations, programmatic offices, or telecommunications programs.

Work Product: Maintain a record of all system interface requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the system interface requirements in the Project File.

Sample System Interface Requirement: *The application must extract records with the following position status indicators from the HRIS mainframe application: EN, X, D, P, T or NW.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the system interface requirements.

Task: **4.3.6**
Define Communication Requirements

Description: The communication requirements define connectivity and access requirements within and between user locations and between other groups and applications.

The following factors should be considered when defining communication requirements.

- Communication needs of the user and customer organizations.
- User organization's existing and planned telecommunications environment (e.g., LANs, WANs, and dial-up).
- Projected changes to the current communication architecture, such as the connection of additional local and remote sites.
- Limitations placed on communications by existing hardware and software including:
  - existing user systems
  - existing applications that will interface with the software product
  - existing organizations that will interface with the software product
- Organization, Government, and industry standards that define communication requirements and limitations

Work Product: Maintain a record of all communication requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the communication requirements in the Project File.

Sample Communication Requirement: *The application must execute online in the organization's local area network environment.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability accuracy, and completeness of the communications requirements.

Task: **4.3.7**
Define Computer Security and Access Requirements

Description: Develop the computer security requirements in conjunction with the system owner's Computer System Security Officer (CSSO) or the Assistant Computer Protection Program Manager (ACPPM). This involvement affords early determination of classifications and levels of protection required for the software product.

If a software product under development processes sensitive personal information, appropriate safeguards must be established to protect the information from accidental disclosure.

Implement applicable security procedures to assure data integrity and protection from unauthorized disclosure, particularly during development efforts. The organization that owns the data defines the data classification. The project team must be aware of all the types of data and of any classified or proprietary algorithms used in the software product.

Procedure: Use the following procedure to determine computer security requirements.

1. Identify the types of data that will be processed by the software product.

2. Determine preliminary data protection requirements.

    a. For software products processing classified information refer to DOE 5639.6, CLASSIFIED COMPUTER SECURITY PROGRAM, September 15, 1992, attachment III, page III-19, paragraph 5.c.(4) - Applications Software.

    b. For software products processing unclassified information, refer to DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN, dated December 1993.

    c. For software products processing sensitive information refer to Chapter 5 of the DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN.

    d. For software products processing sensitive personal information, contact the Freedom of Information Office for coordination and assistance in complying with DOE 1800.1A, PRIVACY ACT.

Procedure, continued:

e. For software products that are considered to be mission essential refer to paragraph 5.4.1.3 and Chapter 8 of DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN.

3. Coordinate with the owner of the host platform to identify existing supporting computer security controls, if applicable.

4. Incorporate security requirements into the Software Requirements Specification.

Work Product: Maintain a record of all security and access requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the security and access requirements in the Project File.

Sample Access Control Questions: The following list provides sample questions that can be used to help define the access controls for the software product.

- What access restrictions are placed on the users by their organization or programmatic office?
- What are the audit and other checking needs for the software product?
- What separation of duties, supervisory functions related to control, operating environment requirements, or other functions will impact the software product?
- What measures will be used to monitor and maintain the integrity of the software product and the data from the user's viewpoint?

Sample Security Requirement: *The application must maintain a record of all user access attempts sorted by authorized and unauthorized users.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the computer security and access requirements.

References:

- DOE Headquarters Computer Protection Plan (CPP) for Unclassified Systems, February 1995, describes the unclassified computer security program for Headquarters organizations.

- DOE Order 1360.2B, UNCLASSIFIED COMPUTER SECURITY PROGRAM, provides guidance for organizations to implement a computer security program for sensitive information.

- DOE Order 5639.6, CLASSIFIED COMPUTER SECURITY PROGRAM, provides guidance for classified systems.

- DOE 1800.1, PRIVACY ACT

Task: **4.3.8**
Define Backup and Recovery Requirements

Description: Develop the requirements for data backup, recovery, and operation startup for the software product in conjunction with the site authority for continuity of operations. If a software product has been defined as mission essential, a Continuity of Operations Plan must be developed. A checklist is provided in *Exhibit 4.3-1, Checklist for Identifying Mission-Essential Software,* to determine if the software is mission essential.

Work Product: If a software product is determined to be mission essential, a Continuity of Operations Plan must be developed. If the software product is not mission essential, a continuity of operations statement is required. Two samples of continuity of operations statements that are appropriate for software that is not mission essential are provided after the checklist. Place a copy of the Continuity of Operations Statement or Plan in the Project File.

Review Process: Conduct structured walkthroughs as needed to assure the necessity, testability, accuracy, and completeness of the backup and recovery requirements.

Reference: *Disaster Recovery Program Guidelines;* Department of Energy; Office of Information Resource Management; Policy, Plans, and Oversight, July 1991.

Exhibit 4.3-1. Checklist for Identifying Mission-Essential Software

The checklist is intended to be used to help identify software products that are mission essential. If a "yes" answer is selected for one or more of the criteria, the software product is mission essential and a Continuity of Operations Plan must be developed.

	Criterion	Yes	No
1	Inability to perform function adversely affects national security.		
2	Inability to perform function adversely affects safety of individuals.		
3	Needed for military effort and civil defense activities during a national emergency.		
4	Needed for mobilization and protection of material and manpower during national emergency.		
5	Function required for maintenance of public health, safety, and order.		
6	Maintains records essential to preservation of legal rights.		
7	Large financial loss incurred with inability to perform functions.		
8	Large expense incurred if performing function by other means.		
9	Primary repository of information reported to Congress or other agencies.		
10	Critical for compliance with federal regulatory requirements.		
11	Sole source of data unobtainable by other means, or not easily recreated.		

Mainframe Sample Statement:

The backup and recovery of the Human Resources Information System (HRIS) comes under the umbrella of the Virtual Machine system backup and recovery procedure. The Virtual Machine system is backed up daily by VMBACKUP under the control of VMSCHEDULE, an automatic job scheduler software package. It is possible to reconstruct HRIS to its state just prior to any system crash by restoring the data base using VMBLIST, another software package. Additionally, HRIS has a software utility called CALLADM, which, with the appropriate parameters, can provide backup and restore capabilities at the directory record level. This utility is fully documented in Appendix F of the HRIS Installation and Reference Guide. The Virtual Machine system is backed up once a week and the tapes are vaulted offsite.

Microcomputer Sample Statement:

The Human Resources Information System (HRIS) is backed up daily by the HRIS data base administrator, using HRIS system utilities. Both the data base and the HRIS log files are backed up. If there is a media failure, it is possible to reconstruct the data base to its state as of the most recent backup of the log files, using an automated procedure. It is possible to perform backups while HRIS users are connected and transactions are in progress, as well as when they are not. HRIS has a Continuity of Operations Plan that allows its users to continue operations using a server in Germantown, Maryland should the local area network (LAN) become unavailable for a significant period. HRIS will be included in the Continuity of Operations Plan being developed for the Human Resources and Administration LAN. This Continuity of Operations Plan is scheduled for completion during the current fiscal year.

Task: **4.3.9**
Define Data Requirements

Description: Data requirements identify the data elements and logical data groupings that will be stored and processed by the software product. The identification and grouping of data begins during the Requirements Definition Stage and is expanded in subsequent stages as more information about the data is known.

Work Product: The major output of the data requirements identification process is a data dictionary. A data dictionary provides an ordered set of definitions about data inputs and outputs, and data stores. In the Requirements Definition Stage, the data dictionary contains a minimum amount of information about data elements such as definitions of the entities, how the data are stored, and data flows to or from other applications. The data dictionary is refined during the design stages as data elements are documented in more detail, and the logical groupings of data elements are formed into interrelated tables or record descriptions.

Work Product: Maintain a record of all data requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the data requirements in the Project File.

Sample Data Requirement: *Records imported from HRIS and ABCD must be matched to the application using the employee's social security number.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the data requirements.

Task: **4.3.10**
Define Implementation Requirements

Description: Describe the requirements anticipated for implementing the software product (e.g., user production cycle). The high-level implementation requirements are identified early in the lifecycle to support decisions that need to be made for the software engineering approach. The implementation requirements are expanded into a full implementation approach during the design stages.

The following paragraphs provide highlights of some of the implementation requirements that need to be considered.

Operating Environment: Identify any capacity restrictions on the existing hardware or software that needs to be addressed and identify any hardware or software that needs to be acquired (e.g., communication hardware, file servers, off-the-shelf software, network interface cards, and LAN utilities). If hardware or software must be acquired, identify the necessary acquisition activities. These activities include preparing specifications, estimating costs, scheduling procurement activities, selection, installation, and testing.

Conversion: Identify requirements for converting data from an existing or external application to the new software product. Consider requirements for data entry, data protection, computer time, conversion programs, personnel, and other resources that will be needed. Also identify the requirements for the conversion of software, if necessary. Implementing a new application may involve converting software from one environment to another, or modifying software to interface with other applications. Include requirements for testing the conversion process and validating that it was successfully accomplished.

Installation: Identify the installation requirements for any new hardware, operating system, or software. For hardware installations, consider environmental factors such as air conditioning, power supply, and security requirements. For software installations, consider proprietary software such as data base management systems. For application software, consider the installation of the application's programs, parallel operation of the old and new applications, or the cutover from a test to a production environment. Hardware and software installation must be coordinated with the work cycles of the user organization to create a minimum of disruption, and to assure that data are available as needed. Installation must be scheduled to assure that, when data conversion is necessary, the needed data are protected.

Training: Identify the specific training needs for various categories of users and administrators. Also identify training requirements for personnel time, computer time, training facilities, and training data base(s).

Documentation: Identify requirements for the development and distribution of operational documentation for software support personnel and user documentation. Operational documentation may include job control procedures and listings, operational instructions, system administration responsibilities, archiving procedures, and error recovery. User documentation includes the users manual, step-by-step instructions, online documentation, and online help facilities.

Work Product: Maintain a record of all implementation requirements. Save for incorporation into the Software Requirements Specification. Place a copy of the implementation requirements in the Project File. This information will also be used to develop an Implementation Plan in the Functional Design Stage.

Sample Conversion Requirement: *All Julian dates found in the extract files must be converted to Gregorian dates.*

Review Process: Conduct structured walkthroughs as needed to ensure the necessity, testability, accuracy, and completeness of the implementation requirements.

Activity: **4.4**
Compile Project Requirements

Responsibility: Project Manager/Team

Description: Compile the requirements gathered during the requirements analysis process in preparation for the development and delivery of the draft Software Requirements Specification. The following steps should be performed as part of the requirements compilation activity.

- Select and use a standard format for describing the requirements.
- Present the logical and physical requirements without dictating a physical design or technical solutions.
- Write the requirements in nontechnical language that can be fully understood by the system owner and users.
- Organize the requirements into meaningful groupings (e.g., all security-related requirements or all requirements for generating reports).
- Develop a numbering scheme for the unique identification of each requirement.
- Select a method for: (1) tracing the requirements back to the sources of information used in deriving the requirements (e.g., specific system owner/user project objectives); and (2) threading requirements through all subsequent lifecycle activities (e.g., testing).

Tasks: The following tasks are involved in the compilation of the project requirements.

4.4.1 Develop Requirements Traceability Matrix
4.4.2 Develop Software Requirements Specification

Task: **4.4.1**
Develop Requirements Traceability Matrix

Description: A requirements traceability matrix is a table used to trace project lifecycle activities and work products to the project requirements.

Every project requirement must be traceable back to a specific project objective(s) described in the Project Plan. This traceability assures that the product will meet all of the project objectives and will not include inappropriate or extraneous functionality.

All work products developed during the design, code, and testing processes in subsequent lifecycle stages must be traced back to the project requirements described in the Software Requirements Specification. This traceability assures that the product will satisfy all of the requirements and remain within the project scope.

It is also important to know the source of each requirement, so that the requirements can be verified as necessary, accurate, and complete. Meeting conference records, user survey responses, and business documents are typical sources for project requirements.

Work Product: Develop a matrix to trace the requirements back to the project objectives identified in the Project Plan and forward through the remainder of the project lifecycle stages. Place a copy of the matrix in the Project File. Expand the matrix in each stage to show traceability of work products to the requirements and vice versa.

Sample Traceability Matrix: One method for tracing requirements is a threading matrix that groups requirements by project objectives. Under each project objective, the source of the requirement, the unique requirement identification number, and the lifecycle activities are listed in columns along the top and the project requirements in rows along the left side. As the project progresses through the lifecycle stages, a reference to each requirement is entered in the cell corresponding to the appropriate lifecycle activity. *Exhibit 4.4-1, Sample Requirements Traceability Matrix,* provides a sample matrix format.

Exhibit 4.4-1. Sample Requirements Traceability Matrix

Requirement	Source of Requirement	Unique Number	Design Spec.	Program Module	Test Spec.
Objective 1: Security					
The software product shall have four user access levels with the capability to add new access levels in the future.	conference record dated 5/19/95	SYSADM 1.0			
Each user access level shall have a unique designation.	conference record dated 5/19/95	SYSADM 1.1			
One user access level shall allow read-only access to the production data base.	conference record dated 5/19/95	SYSADM 1.2			
The second user access level shall allow read and write access to the production data base.	conference record dated 5/19/95	SYSADM 1.3			
The third user access level shall allow read, write, and delete access to the production data base and read-only access to the history data base.	conference record dated 5/19/95	SYSADM 1.4			
The fourth user access level shall allow read, write, and delete access to all application data bases.	conference record dated 5/19/95	SYSADM 1.5			

Task: **4.4.2**
Develop Software Requirements Specification

Description: The Software Requirements Specification describes the inputs to be supplied by the user or other sources, the processing that needs to occur, and the outputs desired by the user or required by interfacing systems. The emphasis should be placed on specifying product functions without implying how the product will provide those functions. This approach provides maximum flexibility for the product designers. The how-to of product implementation is determined in the design stages.

Work Product: Prepare the Software Requirements Specification by integrating all of the requirements developed during this stage. Several formats are available for organizing the requirements information (e.g., from a functional perspective or a data processing perspective).

Document all design constraints including processing, performance, interface, resource, safety, security and reliability requirements. Define data constraints such as limits, formats, messages, commands, and displays.

Review Process: Conduct structured walkthroughs as needed to ensure that the Software Requirements Specification is accurate, complete, and expresses the requirements in a manner that can be understood by the system owner.

The completion of the draft Software Requirements Specification is an appropriate time to schedule an In-Stage Assessment (ISA). The *In-Stage Assessment Process Guide* provides a description and instructions for conducting an ISA. A copy of the guide is provided in Appendix D.

Activity: **4.5**
Establish Functional Baseline

Responsibility: Project Manager/Team

Description: The functional baseline, sometimes called a system requirements baseline, is the main technical work product of the Requirements Definition Stage. The system requirements are baselined after the system owner's formal approval of the Software Requirements Specification. Once the requirements are baselined, any changes to the requirements must be managed under change control procedures established in the Software Configuration Management Plan. Approved changes must be incorporated into the Software Requirements Specification.

Work Product: Prepare the final Software Requirements Specification and submit to the system owner and users for their review and approval. The approved Software Requirements Specification is the official agreement and authorization to use the requirements for the software product design. Approval implies that the requirements are understood, complete, accurate, and ready to be used as the basis for the subsequent lifecycle stages.

It is important for the system owner/users to understand that changes to the approved Software Requirements Specification affect the project scope and therefore can change the project cost, resources, or schedule. It is the responsibility of the project manager and project team to identify system owner/user requested changes that would result in a change of project scope; evaluate the potential impact to the project costs, resources, or schedule; and notify the system owner of the project planning revisions that will be required to accommodate their change requests.

Place a copy of the approved Software Requirements Specification in the Project File.

Review Process: The Software Requirements Specification should be reviewed by the system owner and users. After making the changes needed to resolve problems found during the review, the functional baseline is formally established upon receipt of the system owner's approval.

Activity: **4.6**
Develop Project Test Plan

Responsibility: Project Manager

Description: The Project Test Plan is a narrative and tabular description of the test activities planned for the project. The Project Test Plan should establish the testing necessary to validate that the project requirements have been met and that the deliverables are at an acceptable level in accordance with existing standards. The plan also ensures that a systematic approach to testing is established and that the testing is adequate to verify the functionality of the software product.

The Project Test Plan includes the resources, project team responsibilities, and management techniques needed to plan, develop, and implement the testing activities that will occur throughout the lifecycle. If individuals outside of the project team perform system and acceptance testing, the plan includes the responsibilities and relationships of external test groups.

In this stage, the plan is written at a high level and focuses on identifying test methodologies and test phases. Detailed information about test products (i.e., test plans, test procedures, and test reports) is added to the Project Test Plan as the project progresses through subsequent lifecycle stages.

Development of the Project Test Plan is the responsibility of the project manager. If a test group outside the project team will be involved in any test phase, the project manager must coordinate the Project Test Plan with each test group.

The Project Test Plan must be reviewed and approved by the system owner prior to conducting any tests.

Preparation of the Project Test Plan involves the following tasks.

4.6.1 Identify Test Methodologies
4.6.2 Identify Test Phases
4.6.3 Identify Test Environment Requirements

Note: For small software projects, a formal Project Test Plan may not be necessary; however a test approach and testing are required.

Work Product: When the Project Test Plan is complete, it should contain the following information:

- Describe the occurrence and timing of the test phases in the lifecycle and the entrance and exit criteria for each test phase.
- Specify the test products at each test phase. Describe the types and scope of the testing activities to be performed on each component of the application and the group who is responsible to produce them.
- Map what requirements are verified in what test phase.
- Establish the criteria for evaluating the test results of each test phase.
- Make an initial determination of the resources necessary to accomplish the testing.
- Identify the appropriate person or group to conduct each type of testing activity.
- Outline the test environment (hardware, software, test tools, and data) needed to conduct the tests.
- Develop a preliminary schedule for executing the test activities.

Place a copy of the Project Test Plan in the Project File.

Review Process: Conduct structured walkthroughs to assure the Project Test Plan document adequately describes all testing activities, test schedules, test products, test responsibilities, the testing methodology, and the required resources.

Task: **4.6.1**
Identify Test Methodologies

Description: The Project Test Plan should specify the testing methodologies planned for the project including the types of tests required, test documents, test methods, and test data collection. Each test from unit through acceptance testing is specified in terms of entrance and exit criteria and the expected level of involvement from the project team, test group, and other functional areas.

Unit and integration tests with appropriate data must be developed to exercise and validate all specified application requirements, functions, and objectives. System and Acceptance tests validate that the integrated system meets the requirements.

Each type of test must use controlled computer generated or live data as specified. The test data must be prepared to include values that will verify the functional capabilities of the software test component, identify its limitations and deficiencies (if any), exercise its capabilities, and verify that the software component performs its intended function as required.

If pilot testing or a phased implementation is required for the software product, the Project Test Plan should include such requirements. In the case of an implementation involving phased software releases, the plan should include the requirements for regression testing of the complete application as new elements are introduced.

For each type of test conducted, the test results are compared with the expected results. Discrepancies are identified and any problems resolved. Retesting is required to verify that the problem solution eliminates the problem and does not introduce new errors. The final test results are accompanied by a completed test results/error log form. This form is completed by the individual(s) responsible for testing and attached to the documents that certify the completion of each type of test.

Task: **4.6.2**
Identify Test Phases

Description: The software product should be tested in four sequential phases: unit, integration, system, and acceptance. Some projects may require additional types of tests (such as prototype testing for offsite installations). The four test phases and prototype testing are described below.

Unit Test Phase: The unit test phase involves testing of the individual software units or groups of related units. A unit is a component that is not subdivided into other components; it is a logically separable part of a computer program. Evaluate each unit of code on how well it meets the performance requirements for which it was designed. Consider timing, memory, accuracy in producing numerical and logical results; and the preparation of input and output required for validating program logic, syntax, and performance requirements. This test phase is performed by the programmer(s) responsible for writing the code.

Integration Test Phase: Integration testing is an orderly progression of testing in which software elements, hardware elements, or both are combined and tested to evaluate the interaction between them. Each program/module must be tested. Integration testing is required to validate that groups of related programs, when combined to establish an integrated functional module of code, interface properly, and perform the software functions for which they were designed. Examine the source program/module statements to ensure that the program logic meets the requirements of the design and that the application satisfies an explicit functional requirement. This test phase is performed by the project team.

System Test Phase: The system test phase tests the integrated hardware and software to verify that the software product meets its specified requirements and operates successfully on the host platform. This test phase is required to validate, when the entire software product is loaded onto the host platform, that the proper initialization is performed; decision branching paths are appropriate; and all software functions are performed as specified in the Software Requirements Specification. System testing validates that the software product produces the required outputs and interfaces properly with other systems with which the software product gives or receives data; that transaction response times meet user expectations; and machine resource allocation and utilization are within expected norms. This test phase can be performed by the project team or by an independent test group with support from the project team.

Acceptance Test Phase: Acceptance testing is conducted to determine whether a software product satisfies its acceptance criteria and to enable the system owner's organization to determine whether to accept the software product. The acceptance test is required to validate that the software, its related documentation, tools, and hardware, satisfy all of the specified requirements and objectives of the system owner's organization, DOE standards, the requirements specification, and the design criteria. Acceptance testing will include tests of all intrasystem interfaces; and the use of all manuals, documentation, procedures, and controls. This test phase can be performed by the project team with system owner and user observers or by system owner and user representatives with support from the project team.

Prototype Testing: In addition to the four test phases, a prototype or site test can be used when software must be physically transported, installed, and made operational at a computer facility other than at the site(s) where the acceptance test was conducted. When required, this test is conducted at selected user location(s) that will totally test the software product under "live" conditions with users and support personnel.

Task: **4.6.3**
Identify Test Environment Requirements

Description: The Project Test Plan should outline what is needed to perform testing activities throughout the project lifecycle including personnel, hardware, software, space, and other environmental requirements. As much testing as possible should be performed on the same equipment that will be used for the production system. In many cases, this information is not fully known until the System Design Stage.

The following are some of the considerations for test environment requirements.

- Evaluate automated testing tools for the following:
  - Generation of test scripts
  - Creation of result and error repositories
  - Consideration of each tool's benefits and costs
  - Use of simulators
- Determine local area network, wide area network, and metropolitan area network testing environment(s), as needed
- Determine test lab, data generation, and error correction support
- Identify Beta test sites

Activity: **4.7**
Develop Acceptance Test Plan

Responsibility: Project Team

Description: The Acceptance Test Plan is a description of the test activities planned for project acceptance. The Acceptance Test Plan should establish the testing necessary to validate that the project requirements have been met and that the deliverables are at an acceptable level in accordance with existing standards. The plan also assures that a systematic approach to acceptance testing is established and that the testing is adequate to verify the functionality of the software product.

The complete set of system requirements and acceptance criteria form the basis for determining the overall approach to acceptance testing and the specific testing and examination methods. Features of the installation site and the software system affect how the software acceptance testing will be done. Unique arrangements may be necessary when the software cannot be completely installed and executed in a live environment. Multiple configurations may have to be distributed at several installation sites.

When a new system is a replacement for one already in use, the acceptance test must assure the integrity of the users business operations while placing the replacement into operation. For example, the old system and the new system are used in parallel until complete functionality has been verified. In some cases, the acceptance process may take several months to assure that a complete business or accounting cycle has occurred. This concern will influence the approach to software acceptance testing.

Work Product: Software acceptance testing must be documented carefully with traceability of test cases to the software requirements and acceptance criteria established by the system owner. As a minimum, the acceptance test plan should address the following requirements.

- Identification of the personnel involved in the acceptance test process and their testing responsibilities. If individuals outside of the project team perform acceptance testing, include the responsibilities and relationships of external test groups.

- Traceability of test designs and cases to software requirements.

- The objectives and constraints for each test.

Work Product, continued:

- Complete test cases and test procedures including inputs and expected outputs for each test case.
- Descriptions of error reporting, analysis, and resolution.
- Location(s) where testing will occur, the testing approach, type of facilities, and tester training.
- Acquisition of special purpose testing equipment, tools, and software.
- Resources and cost estimation to accomplish testing.

Place a copy of the draft Acceptance Test Plan in the Project File. The draft plan will be reviewed during the Software Integration and Testing Stage and delivered as a final document.

Review Process: Conduct structured walkthroughs to assure the draft Acceptance Test Plan adequately describes all testing activities, test schedules, test products, test responsibilities, the testing methodology, and the required resources.

Activity: **4.8**
Select Design Methodology

Responsibility: Project Team

Description: A systematic approach for building the functional and system designs for the software product simplifies the process and results in a software product that is testable, reliable, and maintainable. A complete design methodology includes the following elements:

- A methodology that is compatible with the requirements analysis methodology and any automated tools used by the project team.
- Straightforward rules that relate information obtained during requirements analysis to a distinct software structure.
- Design standards that comply with the site's current software engineering practices, the system owner organization's standards, and the constraints imposed by the software and hardware tools used by the project team.
- A practical approach to design that is amenable to a wide variety of software products.
- The development of small, intermediate design products that can be used to measure quality and progress.
- An evolution process from functional to system design.
- Well-defined measures to assess the quality of the design.
- Guidance on how to detect and correct design features that reduce maintainability and reusability.

The value of a design methodology can be significantly enhanced by automated tools that directly support the methodology. Automated tools provide assistance in generating, maintaining, and analyzing design diagrams and data dictionaries. The use of such tools typically results in a design that is easier to maintain, higher in quality, and more complete than designs produced without automated tools. The increased quality leads to significant productivity gains during software programming and testing.

Sample Design Methods: The following are examples of some common design methodologies.

- Function-oriented design methods model the software product by breaking it into components, identifying the inputs required by those components, and identifying the outputs produced by them. Function-oriented design methods include structured analysis and structured design. The major models or design representations used by this method are data flow diagrams, data dictionaries, structure charts, and process specifications.

- Data-oriented design methods use program structures that are derived from the data structures. Tree diagrams are typically used to represent both the data and the program structures.

- Object-oriented design methods produce a software architecture based on the objects manipulated by systems or subsystems rather than by functions. An object-oriented design closely resembles a model of reality since it captures the real-world objects and the operations taken by or upon them. The design structure tends to be layers of abstraction where each layer represents a collection of objects with limited visibility to other layers.

Work Product: Create a description of the design methodology and distribute it to the project team, system owner, and users. Place a copy of the design methodology description in the Project File.

Review Process: Conduct a structured walkthrough to verify that the design methodology is appropriate for the scope and objectives of the project. A structured walkthrough is not needed when the methodology has been used successfully on similar projects for the same system owner/user computing environment.

Activity: **4.9**
Revise Project Plan

Responsibility: Project Manager

Description: Once the requirements are baselined, determine if the project estimates for resources, cost, and schedule need to be revised and if the selected development approach is still appropriate for the size and complexity of the project.

Work Product: Review the Project Plan for accuracy and completeness of all Requirements Definition Stage activities and make any changes needed to update the information. Expand the information for the Functional Design Stage to reflect accurate estimates of resources, costs, and hours.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Review Process: Conduct a structured walkthrough to assure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the Functional Design Stage.

The Project Plan is formally reviewed during the In-Stage Assessment and Stage Exit processes.

Activity: **4.10**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the Requirements Definition Stage Exit process. Additional ISAs can be performed during the stage, as appropriate. An ISA is recommended after the completion of the Software Requirements Specification.

Provide the reviewer with copies of all work products developed or revised during the Requirements Definition Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/best practices.
- Sound project management practices are being used.
- The project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide.* A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **4.11**
Conduct Requirements Definition Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the Requirements Definition Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the *Stage Exit Process Guide* is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the Functional Design Stage.

Software Engineering Methodology

Chapter 5.0
Functional Design Stage

Table of Contents

Chapter **Page**

5.0 Functional Design Stage 5.0-1
- 5.1 Determine Software Structure 5.1-1
  - 5.1.1 Identify Design Entities 5.1-2
  - 5.1.2 Identify Design Dependencies 5.1-4
- 5.2 Design Content of System Inputs and Outputs 5.2-1
- 5.3 Design User Interface 5.3-1
  - 5.3.1 Design Menu Hierarchy 5.3-3
  - 5.3.2 Design Data Entry Screens 5.3-5
  - 5.3.3 Design Display Screens 5.3-6
  - 5.3.4 Design Online Help 5.3-8
  - 5.3.5 Design System Messages 5.3-10
- 5.4 Design System Interfaces 5.4-1
- 5.5 Design System Security Controls 5.5-1
- 5.6 Build Logical Model 5.6-1
- 5.7 Build Data Model 5.7-1
- 5.8 Develop Functional Design 5.8-1
  - 5.8.1 Develop Functional Design Document 5.8-2
  - 5.8.2 Conduct Functional Design Review 5.8-3
- 5.9 Initiate Procurement of Hardware and Software 5.9-1
- 5.10 Revise Project Plan 5.10-1
- 5.11 Conduct In-Stage Assessment 5.11-1
- 5.12 Conduct Functional Design Stage Exit 5.12-1

Chapter: **5.0**
Functional Design Stage

Description: The functional design process maps the "what to do" of the Software Requirements Specification into the "how to do it" of the design specifications. During this stage, the overall structure of the software product is defined from a functional viewpoint. The functional design describes the logical system flow, data organization, system inputs and outputs, processing rules, and operational characteristics of the software product from the user's point of view. The functional design is not concerned with the software or hardware that will support the operation of the software product, or the physical organization of the data or the programs that will accept the input data, execute the processing rules, and produce the required output.

The focus is on the functions and structure of the components that comprise the software product. The goal of this stage is to define and document the functions of the software product to the extent necessary to obtain the system owner and users understanding and approval and to the level of detail necessary to build the system design.

Prototyping of system functions can be helpful in communicating the design specifications to the system owner and users. Prototypes can be used to simulate one function, a module, or the entire software product. Prototyping is also useful in the transition from the functional design to the system design.

Input: The following work products provide input to this stage.

- Project File
- Software Configuration Management Plan *(draft)*
- Continuity of Operations Statement/Plan
- Data Dictionary
- Requirements Traceability Matrix
- Software Requirements Specification
- Project Test Plan
- Acceptance Test Plan *(revised)*
- Design methodology
- Project Plan *(revised)*
- Software Quality Assurance Plan

High-Level

Activities: The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage. These activities represent the minimum requirements for a large software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

5.1 Determine Software Structure
5.2 Design Content of System Inputs and Outputs
5.3 Design User Interface
5.4 Design System Interfaces
5.5 Design System Security Controls
5.6 Build Logical Model
5.7 Build Data Model
5.8 Develop Functional Design
5.9 Initiate Procurement of Hardware and Software
5.10 Revise Project Plan
5.11 Conduct In-Stage Assessment
5.12 Conduct Functional Design Stage Exit

Output: Several work products are developed during this stage. The work products listed below are the minimum requirements for a large software project. Deviations in the context and delivery of these work products are determined by the size and complexity of a project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

- Design records
- Logical model
- Data Dictionary
- Requirements Traceability Matrix *(expanded)*
- Functional Design Document
- Minutes from Functional Design Review
- Hardware and software procurement records
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 5.0-1, Functional Design Stage Activities and Work Products by Project Size.* The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large software products.

Review Process: Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. In addition, a Preliminary Design Review will be conducted. This review is an important milestone in the design process. The time and resources needed to conduct the walkthroughs and Functional Design Review should be reflected in the project resources, schedule, and work breakdown structure.

Reference: *Appendix C, Conducting Structured Walkthroughs*, provides a procedure and sample forms that can be used for structured walkthroughs.

Exhibit 5.0-1. Functional Design Stage Activities and Work Products by Project Size

Work Activity		Project Size L	Project Size M	Project Size S	Work Product	Scheduled Deliverables L	Scheduled Deliverables M	Scheduled Deliverables S
5.1	Determine Software Structure	R	R	R	Design entities and dependencies	I	I	I
5.2	Design Content of System Inputs and Outputs	R	R	R	System input and output design	I	I	I
5.3	Design User Interface	R[2]	R[2]	R[2]	User interface design	I[2]	I[2]	I[2]
5.4	Design System Interfaces	R	R	R	System interface design	I	I	I
5.5	Design System Security Controls	R	R	R	System security control design	I	I	I
5.6	Build Logical Model	R	R	R	Logical model	R	R	R
5.7	Build Data Model	R	R	R	Data dictionary *(revised)*	R	R	R
5.8	Develop Functional Design	R	R	R	Requirements Traceability Matrix *(expanded)* Functional Design Document Functional Design Review minutes	R R R	R R R	R R R
5.9	Initiate Procurement of Hardware and Software	A	A	A	Procurement records	N	N	N
5.1	Revise Project Plan	R	R	A	Project Plan *(revised)*	R	R	A
5.11	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
5.12	Conduct Functional Design Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
S = Small

Minimum Requirements: R = Required
A = As Appropriate
N = Not Appropriate

I = Input to other deliverables
[1] = Completed by reviewer
[2] = Can use existing plan/procedure

Bibliography: The following materials were used in the preparation of the Functional Design Stage chapter.

1. Barker, Richard, *CASE*METHOD,* Tasks and Deliverables, 1990. pp. 5-10 and 5-11.

2. Bramucci, Wilma, "Systems Development Software," *Faulkner Technical Reports, Inc.*, June 1989. pp. 1-7.

3. Gane, Chris and Sarson, Trish, *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, 1979.

4. *Software Engineering Handbook*, Chapter 4, Software Requirements Analysis. pp. 4-1 through 4-9.

5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications,* ANSI/IEEE Std 830-1984, New York, 1984.

6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Design Descriptions,* IEEE Std 1016.1-1993, New York, 1993.

7. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.

8. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans,* ANSI/IEEE Std 1012-1986, New York, 1986.

9. U.S. Department of Commerce, National Bureau of Standards, *Guideline for Planning and Management of Database Applications,* Federal Information Processing Standards Publication 77, 1980. pp. 10-23.

10. U.S. Department of Defense, Military Standard, *Technical Reviews and Audits for Systems, Equipments, and Computer Software,* MIL-STD-1521 B, 1985. pp.5, 33-52.

11. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual,* 1988.

12. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual,* 1988.

13. U.S. Social Security Administration, Office of Systems, *Software Engineering Technology (SET) Manual,* Volume 1, 1990.

14. Yourdon, inc., *Management Overview of Real-Time Structured Analysis and Design*, Edition 3.0, January 1984.

Activity: **5.1**
Determine Software Structure

Responsibility: Project Team Analysts

Description: A hierarchical approach is useful for determining the structure and components of the software product. Software system decomposition is one hierarchical approach that divides the software system into different levels of abstraction. Decomposition is an iterative process that continues until single purpose components (i.e., design entities or objects) can be identified. Decomposition is used to understand how the software product will be structured, and the purpose and function of each entity or object.

The goal of the decomposition is to create a highly cohesive, loosely coupled, and readily adapted design. A design exhibits a high degree of cohesion if each design entity in the program unit is essential for that unit to achieve its purpose. A loosely coupled design is composed of program units that are independent or almost independent.

Several reliable methods exist for performing system decomposition. Select a method that enables the design of simple, independent entities. Functional and object-oriented design are two common approaches to decomposition. These approaches are not mutually exclusive. Each may be applicable at different times in the design process.

The software system decomposition activity includes the following tasks.

5.1.1 Identify Design Entities
5.1.2 Identify Design Dependencies

Task: **5.1.1**
Identify Design Entities

Description: Design entities result from a decomposition of the software product requirements. A design entity is an element (or object) of a design that is structurally and functionally distinct from other elements and is separately named and referenced. The number and type of entities required to partition a design are dependent on a number of factors, such as the complexity of the software product, the design method used, and the programming environment. The objective of design entities is to divide the software product into separate components that can be coded, implemented, changed, and tested with minimal effect on other entities.

Attributes: A design entity attribute is a characteristic or property of a design entity. It provides a statement of fact about an entity. The following are common attributes that should be considered for each design entity.

- Assign a unique name to each entity.
- Classify each entity into a specific type. The type may describe the nature of the entity, such as a subprogram or module; or a class of entities dealing with a particular type of information.
- Describe the purpose or rationale for each entity. Include the specific functional and performance requirements for which the entity was created.
- Describe the function to be performed by each entity. Include the transformation applied to inputs by the entity to produce the desired output.
- Identify all of the external resources that are needed by an entity to perform its function.
- Specify the processing rules each entity will follow to achieve its function. Include the algorithm used by the entity to perform a specific task and contingency actions in case expected processing events do not occur.

Attributes, continued:

- Describe the data elements internal to each entity. Include information such as the method of representation, format, and the initial and acceptable values of internal data. This description may be provided in the data dictionary.

Work Product: Maintain a record of all design entities. The records will be integrated into the Functional Design Document. Place a copy of the design entity information in the Project File.

Review Process: Schedule structured walkthroughs to verify that the design entities are correct, complete, and possess the required attributes.

Task: **5.1.2**
Identify Design Dependencies

Description: Design dependencies describe the relationships or interactions between design entities at the module, process, and data levels. These interactions may involve the initiation, order of execution, data sharing, creation, duplication, use, storage, or destruction of entities.

Identify the dependent entities of the software system design, describe their coupling, and identify the resources required for the entities to perform their function. Also define the strategies for interactions among design entities and provide the information needed to perceive how, why, where, and at what level actions occur.

Dependency descriptions should provide an overall picture of how the software product will work. Data flow diagrams, structure charts, and transaction diagrams are useful for showing the relationship among design entities.

The dependency descriptions may be useful in producing the system integration plan by identifying the entities that are needed by other entities and that must be developed first. Dependency descriptions can also be used to aid in the production of integration test cases.

Work Product: Add specific dependency information to the design entity records. The records will be integrated into the Functional Design Document. Place a copy of the dependency information in the Project File.

Review Process: Schedule structured walkthroughs to verify that the design dependencies are correct and complete.

Activity: **5.2**
Design Content of System Inputs and Outputs

Responsibility: Project Team Analysts

Description: Design the content and format for each of the software product inputs and outputs based on the system input and output requirements identified during the Requirements Definition Stage. Involve the system owner and users in the design process to make certain that their needs and expectations are being met.

Procedure: Use the following procedure to implement the design process.

- Identify the types of electronic and printed input that will be accepted by the software product, such as data entered from source documents and files or records extracted from other systems.
- Identify the types of electronic and printed output that will be produced by the software product; such as data, records, or files; screen displays; and printed reports. Also identify the output that will be exported to other systems.
- Identify the specific input and output items that already exist and the items that will be created as part of the software product.
- Assign a name to each type of input and output and describe each item from a functional perspective.
- Identify the owner/originator of each type of input and output.
- Identify the frequency of each type of input and output.
- Design the content and format for each new input and output item or modify the format of existing items that must be changed to accommodate the new software product.

Work Product: Document the design for the system inputs and outputs in accordance with the project design standards. Discuss the designs with the system owner and users and submit completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the system input and output designs in the Project File.

Review Process: Schedule a structured walkthrough to verify that the system input and output designs are correct and complete.

Activity: **5.3**
Design User Interface

Responsibility: Project Team Analysts

Description: Design a user interface that is appropriate for the users, content, and operating environment for the software product. Determine interface levels for all categories of users. For interactive user environments, prototype the user interface. Arrange for users to experiment with the prototypes so that design weaknesses in the interface can be identified and resolved early. Use prototypes to gain user acceptance of the interface.

If the site or system owner's organization has an existing user interface standard, this standard should be used to specify the user interface for every software product developed for that organization. A user interface standard should be developed and maintained for each organization that does not have one.

Review the standard each time a new software product is planned to verify that the user interface is compatible with the software product's selected system architecture. For example, some DOS-based user interface standards would not be appropriate for a Windows-based software product.

The following tasks are involved in specifying the user interface.

5.3.1 Design Menu Hierarchy
5.3.2 Design Data Entry Screens
5.3.3 Design Display Screens
5.3.4 Design Online Help
5.3.5 Design System Messages

Basic Principles: The following basic principles can help improve the software product user interface when there is graphical, command-based, menu-driven, or block mode features.

- Give users control. Let them choose actions to perform.
- Give users feedback and progress reports. Tell them when the system is working and when an action is completed.
- Make sure programs, windows, and functions are consistent within and with other components of the software product.

Basic Principles,

continued:

- Be consistent in the format and wording of text.
- Keep it simple. White space is as important on the screen as on the printed page. Reduce screen clutter.
- Use special effects carefully and sparingly. Be sure color screens also work in one color--some users are colorblind, and some users have monochrome monitors. Use color consistently. Beeps and other sounds can be annoying; so let users turn sound off.
- Put information where it can be easily seen; avoid information in corners or borders.
- Limit the amount of information users must know. Offer choices instead of making users remember and enter information. Provide defaults, and make sure they are logical and satisfy a large number of users.
- Offer shortcuts. Keyboard shortcuts (e.g., hot keys) and command abbreviations help experienced users work more quickly.
- Help users get out of trouble. Provide messages that are understandable and that offer solutions.
- Let users reverse their actions. If an action will destroy something, identify the object of destruction and wait for a response.

Task: **5.3.1**
Design Menu Hierarchy

Description: Use the following guidelines to improve the design of menu hierarchies.

- Choose an organizing principle for the menu options, such as:

  - Expected frequency of use
  - Logical sequence of operations
  - Alphabetical order (should be used for horizontal word menus with five or more words)

- Put a meaningful title at the top of every menu.

- For full-screen menus, provide symmetric balance by centering the title and the menu options around the center axis of the screen.

- To facilitate scanning, put blank lines between logical groupings of menu options and after about every fifth option in a long list.

- Limit the number of menu choices to one screen.

- Use a menu option selection method that is consistent with the technology available at the user's workstation and the size of the software product being designed, such as:

  - Numbers
  - Letters or letter combinations
  - Cursor movement

- Provide a way for the user to leave the menu without performing any action. Be sure that the option to leave the menu describes the consequences of its selection.

- Words used for menu options should follow these rules:

  - Use words that clearly and specifically describe what the user is selecting.

  - Use common English words rather than computer or technical jargon. When space permits, spell out words completely.

Description, continued:

- Use simple, active verbs to tell users what actions will result from their choice. Try to start each option with a verb.
- Use parallel construction to describe the options.

- Minimize the highlighting used on a menu. Highlighting should be limited to situations where the user needs to know that there is an exception to the normal practice.
- Do not require the user to enter leading or trailing blanks or zeros, and do not include a default value on a menu.
- Display the menu options in mixed letters (i.e., upper and lower case).
- Organize menu hierarchies according to the tasks users will perform, rather than the structure of the software modules.

Work Product: Document the design for the menu hierarchy in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document. Place a copy of the menu hierarchy design in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the menu hierarchy design is complete and logical.

Task: **5.3.2**
Design Data Entry Screens

Description: Use the following guidelines to improve the design of data entry screens.

- When the user must transcribe data directly from a source document to the screen, the layout of the screen should be similar to the layout of the source document.
- Group data fields into logical categories on the screen; provide a header that describes the contents of each category.
- Make areas of the screen that are not needed for data entry or commands inaccessible to the user.
- Do not require the user to enter information that is already available to the software or can be computed by it.
- Do not require the user to enter dimensional units, leading or trailing blanks, or zeros.
- Allow the user to enter data by character replacement.
- Put a caption describing the data to be entered adjacent to each data field; incorporate memory joggers into the caption.
- Justify data entries automatically.
- Display default values in data fields when appropriate.
- Provide context-sensitive help for data entry fields.

Work Product: Document the designs for the data entry screens in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the data entry screen designs in the Project File.

Review Process: Conduct a structured walkthrough to assure that the data entry screen designs are consistent, complete, and logical.

Task: **5.3.3**
Design Display Screens

Description: Use the following guidelines to design display screens that are easy to use and understand.

- Put a title on every display screen. The title should clearly and specifically describe the contents of the screen.
- Display only information that the user needs to know.
- Display data to the user in directly usable form.
- Provide symmetric balance to displays by centering titles and headings and by placing information on both sides of the center axis.
- Every display should indicate how to exit from the screen. Use consistent exit procedures.
- When the display continues over multiple screens, the screen should indicate where the user is in the display (e.g., Screen 1 of 3).
- Consider the skills of the users and the information they will manipulate when information is displayed in multiple windows.
- Data fields need to be grouped into logical categories or according to the structure of a source document (when there is one).
- Be consistent in the use of words and special characters.
- Display text conventionally in mixed letters (i.e., upper and lower case) and with appropriate punctuation. Avoid all uppercase letters. Put a blank line between paragraphs.
- Left justify text, and leave a ragged right margin.
- Avoid hyphenation of words between lines.
- Use abbreviations and acronyms only when they are significantly shorter than the full text and when they will be understood by the user.
- Be consistent with the format of information being displayed.

Table and List

Guidelines: Use the following guidelines to improve the design of online tables and lists.

- Put a meaningful label on the columns and, if appropriate, the rows of tables and lists. Continue the labels when a table or list extends over more than one screen.
- If data items are scrolled, the labels should be fixed on the screen and not be part of the scrolled area (they remain in place as the body of the table or list changes).
- If data items are continued on subsequent screens, the labels should be added to each screen.
- Arrange the items in a table or list in some recognizable order to facilitate scanning.
- Put items in a multiple column list in vertical columns that are read from left to right on the screen.
- Left justify columns of alphabetic data; right justify columns of numeric data or align them by the decimal point or other delimiter.
- Insert a blank line after about every fifth row in a long column.
- Insert a minimum of two spaces between the longest item in a column and the beginning of the next column.
- Start with a one (1) not a zero (0) when listed items are labeled by number.

Work Product: Document the design for the display screens in accordance with the project design standards. Discuss the designs with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the display screen designs in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the display screen designs are consistent, complete, and logical.

Task: **5.3.4**
Design Online Help

Description: Online help is typically requested by users when they want to perform a new, complex, or infrequently used procedure, or when they do not know what else to do. The text of online help messages needs to be planned, drafted, and evaluated as carefully as print documentation. In addition, the layout and format of online help must be designed to deal with the special constraints imposed by the video screen.

Use online help to explain concepts, procedures, messages, menu choices, commands, words, function keys, and formats. Work with the users to identify the level of detail needed for online help. Determine whether the users need a one-line message at the bottom of the screen or a full online explanation with successive levels of detail. Effective online help messages tell users what the software product is doing, where they are in the sequence of screens, what options they have selected, and what options are available.

Guidelines: The following guidelines can improve the design of online help.

- Write online help messages in plain English.
  - Straightforward and reads as if it were spoken.
  - Clear, direct, and simple.
  - Effectively organized with a concern for what users need to know.
- Address the user directly as "you"; use the active voice.
- Use simple action verbs to describe procedures. Do not use nouns to replace pronouns, verbs, and adjectives.
- Describe procedures in logical order.
- Avoid computer terms or other jargon, such as:
  - Terms that are unique to the computer profession or to a particular company.
  - Terms that have a common meaning outside of the data processing environment, but a special meaning within it, such as *boot*, *abort*, *default*, and *utility*.

Guidelines, continued:

- Terms that are made up to describe some special function, such as *ungroup* and *dearchive*.

- Avoid humor in online documentation.

- Write in short complete sentences and paragraphs and use proper punctuation.

- Write sentences in the positive or simple negative. Avoid the passive voice and do not use double negatives.

- Use bullets, numbered lists, and tables to make it easier to find the most important information. Leave ample open space.

  - Use bulleted lists to explain options. Whenever a sentence lists options with commas between them, consider breaking up the text into a bulleted list.
  - Use numbered lists to show the steps in a process.
  - Use a table to explain two or more categories of information.

- Use examples to show users what they should enter and what the results will look like.

- Do not expect users to read more than about three screens of help at one time.

- Provide an orientation to the structure of the software product.

- Whenever possible display help text on the screen with the function or task that is being performed.

- Provide a direct route back to the function or task being performed.

Work Product: Document the design for online help in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document. Place a copy of the online help design in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the online help design is consistent, complete, and logical.

Task: **5.3.5**
Design System Messages

Description: System messages are the various types of information that the system provides to the user such as status messages, user prompts, and error messages.

Status Messages: Status messages are important for giving users the feeling they are in control of the software. They tell users what the software is doing, where they are in the sequence of screens, what options they have selected, and what options are available.

User Prompts: Prompts inform the user to type data or commands or to make a simple choice.

- Use prompts to ask the user to make a simple choice or to enter data or commands. Be as specific as possible.
- Include memory aids in the prompt to help users type a response in the proper format and order, initiate infrequently used processes, or clearly identify exceptions to normal practice.
- When defaults are allowed with prompts, indicate clearly which default value will be initiated.

Error Messages: Error messages should allow users to recover from mistakes by making it clear what the mistake was and how to correct it. Error messages need to be specific about why a mistake was made.

- Design the software product to check for obvious errors.
- Be as specific as possible in describing the cause of an error. Do not use error codes.
- Do not assign blame to the user or the software in an error message. Use a neutral tone.
- Whenever possible, the error message should indicate what corrective action the user needs to take.
- Be consistent in the format, wording, and placement of messages.
- Consider describing error messages at more than one level of detail.

Work Product: Document the design for the system messages in accordance with the project design standards. Discuss the designs with the system owner and users and submit the completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the system message designs in the Project File.

Review Process: Conduct a structured walkthrough to ensure that the system message designs are consistent, complete, and logical.

Activity: **5.4**
Design System Interfaces

Responsibility: Project Team Analysts

Description: Develop a design depicting how the software product will interface with other systems based on the system interface requirements identified in the Requirements Definition Stage. Submit the applicable interface designs for review by the system owner or system administrator for each system that will interface with the software product. Any incompatibilities with the interfaces will be identified early in the design process and corrective actions can be initiated to assure each interface is properly designed and coded.

Sample Issues: The following list provides some of the issues that should be considered when designing the system interfaces.

- System inputs and outputs
- Method of interface
- Volume and frequency of data
- Platform of interfacing system
- Format of data
- Automatic or manual initiation of interface
- Need for polling device(s)
- Verification of data exchange
- Validation of data

Work Product: Document the design(s) for the system interfaces in accordance with the project design standards. Discuss the designs with the system owner and users and submit completed designs for their review and approval. The approved designs will be incorporated into the Functional Design Document. Place a copy of the system interface designs in the Project File.

Review Process: Schedule a structured walkthrough to verify that the system interface designs are correct and complete.

Activity: **5.5**
Design System Security Controls

Responsibility: Project Team Analysts and Security Personnel

Description: Design the security controls that will be incorporated into the software product based on the security and access requirements identified during the Requirements Definition Stage. Design the security controls in conjunction with the site or system owner organization's Computer System Security Officer (CSSO) or the Assistant Computer Protection Program Manager (ACPPM).

Procedure: Use the following procedure to implement the design process.

- Identify the users and organizations that will have access to the software product. Indicate what access restrictions they will have. All persons in a work area may not have the same security access level. Measures should be taken to assure that sensitive materials and software requiring protection are not accessed by unauthorized individuals.

- Identify controls for the software product, such as the user identification code for system access and the network access code for the network on which the software product will reside.

- Identify whether access restrictions will be applied at the system, subsystem, transaction, record, or data element levels. Classified information must be protected in accordance with DOE directives.

- Identify physical safeguards required to protect hardware, software, or information from natural hazards and malicious acts.

- Identify communications security requirements.

Work Product: Document the design for the system security controls in accordance with the project design standards. Discuss the design with the system owner and users and submit the completed design for their review and approval. The approved design will be incorporated into the Functional Design Document. Place a copy of the system security control design in the Project File.

Review Process: Schedule a structured walkthrough to verify that the system security controls are correct and complete. Include the CSSO or the ACPPM in the walkthrough.

References:

- DOE 5639.6, CLASSIFIED COMPUTER SECURITY PROGRAM, September 15, 1992.
- DOE HEADQUARTERS UNCLASSIFIED COMPUTER PROTECTION PLAN, February 1995.

Activity: **5.6**
Build Logical Model

Responsibility: Project Team Analysts

Description: The logical model defines the flow of data through the software system and determines a logically consistent structure for the software. Each module that defines a function is identified, interfaces between modules are established, and design constraints and limitations are described. The focus of the logical model is on the real-world problem or need to be solved by the software product.

A logical model has the following characteristics:

- Describes the final sources and destinations of data and control flows crossing the system boundary rather than intermediate handlers of the flows.
- Describes the net transfer of data across the system boundary rather than the details of the data transfer.
- Provides for data stores only when required by an externally imposed time delay.

When building a logical model, the organization of the model should follow the natural organization of the software product's subject matter. The names given to the components of the model should be specific. The connections among the components of the model should be as simple as possible.

Work Product: The logical model should be documented in user terminology and contain sufficient detail to obtain the system owner's and users' understanding and approval. Use data flow diagrams to show the levels of detail necessary to reach a clear, complete picture of the software product processes, data flow, and data stores.

Maintain the logical model and data flow diagrams for incorporation into the Functional Design Document. Place a copy of the logical model and data flow diagrams in the Project File. Keep the logical model and diagrams up-to-date. They will serve as a resource for planning enhancements during the Maintenance Stage, particularly for enhancements involving new functions.

Review Process: Schedule a structured walkthrough to verify that the logical model is correct, logical, and complete.

Activity: **5.7**
Build Data Model

Responsibility: Project Team Analysts

Description: A data model is a representation of a collection of data objects and the relationships among these objects. The data model is used to provide the following functions:

- Transform the business entities into data entities.
- Transform the business rules into data relationships.
- Resolve the many-to many relationships as intersecting data entities.
- Determine a unique identifier (keys) for each data entity.
- Add the attributes (facts) for each data entity.
- Document the integrity rules required in the model.
- Determine the data accesses (navigation) of the model.

Work Product: The data dictionary started in the Requirements Definition Stage is expanded in this stage to catalog every known data element used in the user's work and every system-generated data element. Data elements are documented in detail to include attributes, known constraints, input sources, output destinations, and known formats.

The data dictionary can serve as a central repository of information for both programmers and end users. The dictionary can include business rules, processing statistics, and cross-referencing information for multiple vendor environments.

To expand the data dictionary, define, analyze, and complete data definitions using the following steps.

- Identify data needs associated with various system features.
- Match (verify) data needs with the data dictionary.
- Match the data dictionary with specific data structures.

Work Product, continued:

- Create data record layouts.
- Ensure that all data can be maintained through add, change, or delete functions.

The data dictionary is further refined in the System Design Stage to complete the information on data elements, entities, files, physical characteristics, and data conversion requirements.

Sample Attributes:

The following is a sample of the type of attributes (information) that should be included for each element in a data dictionary.

Long data name (full name)
Short data name (abbreviation)
Alias
Data definition
Owner(s)
Occurrence(s)/key
Program mode
Input source(s); e.g., screens, external interfaces, system generated
Output destination(s); e.g., screens, reports, external interfaces
Values/meanings
Protection/security
Default value
Length/precision
Character set (type)
Format
Range
Surface edits
Remarks

Review Process:

Schedule a structured walkthrough to verify that the data dictionary is correct and complete.

Activity: **5.8**
Develop Functional Design

Responsibility: Project Team

Description: The software functional design describes how the software product will be structured to satisfy the requirements identified in the Software Requirements Specification. It is a description of the software structure, components, interfaces, and data necessary before coding can begin.

The software functional design is a model or representation of the software product that is used primarily for communicating software design information to facilitate analysis, planning, and coding decisions. It represents a partitioning of the software system into design entities and describes the important properties and relationships among those entities. Design descriptions may be produced as documents, graphic representations, formal design languages, records in a data base management system, and CASE tool dictionaries.

Within the functional design, the design entities can be organized and presented in several ways. The goal of this activity is to compile the design entities and their associated attributes in a manner that facilitates the access of design information from various viewpoints (e.g., project management, configuration management, quality assurance, and testing). Also, the design entities and their attributes must be described in terms that are understandable to the system owner and users.

Work Product: Each requirement identified in the Software Requirements Specification must be traceable to one or more design entities. This traceability ensures that the software product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the functional design to the requirements. Place a copy of the expanded matrix in the Project File.

The following tasks are involved in developing the functional design.

5.8.1 Develop Functional Design Document
5.8.2 Conduct Functional Design Review

Task: **5.8.1**
Develop Functional Design Document

Description: The Functional Design Document defines the functions of the system in user terminology and provides a firm foundation for the development of the system design. The Functional Design Document should be written from the system owner/users' perspective. This document provides the owner/users with an opportunity to review and provide input to the software product design before system design work is completed.

Work Product: Prepare a draft Functional Design Document. Use the designs developed for inputs, outputs, user and system interfaces, and security controls as input to this document. Submit the draft document to the system owner and users for their review and approval. After making the changes needed to resolve problems found during the review, the approved Functional Design Document becomes an official agreement and authorization to use the functional design as the basis for developing the system design. Place a copy of the approved Functional Design Document in the Project File.

Review Process: Conduct structured walkthroughs as needed to assure that the Functional Design Document is accurate, complete, and describes the functional design in a manner that can be understood by the system owner and users.

The completion of the draft Functional Design Document is an appropriate time to schedule an In-Stage Assessment (ISA). The *In-Stage Assessment Process Guide* provides a description and instructions for conducting an ISA. A copy of the guide is provided in Appendix D.

Task: **5.8.2**
Conduct Functional Design Review

Description: The Functional Design Review is a formal technical review of the basic design approach. The primary goal of the Functional Design Review is to demonstrate the ability of the software design to satisfy the project requirements. The review should be a series of presentations by the project team to the system owner, users, functional area points-of-contact, and Quality Assurance representative. Vendors may be invited to participate in the Functional Design Review when an off-the-shelf software product or hardware item is being considered for the system architecture.

Conduct the Functional Design Review to perform the following verifications.

- Evaluate the progress, technical adequacy, and risk resolution of the selected design approach. Determine whether the approved design approach is being followed by the project team.
- Evaluate the progress, technical adequacy, and risk resolution of the selected test approach. Review the following items:
  - Organization and responsibilities of group conducting tests
  - Project Test Plan
  - Planned format, content, and distribution of test reports
  - Planned resolution of problems and errors identified during testing
  - Retest procedures
  - Change control and configuration management of test items
  - Special test tools not required as deliverables
- Evaluate the methodology to be used to meet quality assurance requirements.
- Establish the existence and compatibility of the physical and functional interfaces.

Description, continued:

- Determine whether the functional design embodies all of the software product requirements.
- Verify that the design represents software that can meet the functional, data, and interface requirements.
- Review the planned user interfaces to the software. Examples of the types of design information to review:
  - Operating modes for each display station. For each mode, the functions performed, the displays and controls used.
  - The format and content standards for each screen (e.g., data locations, spaces, abbreviations, the number of digits, all special symbols, alert mechanisms).
  - Control and data entry devices and formats (e.g., keyboards, special function keys, and cursor control).
  - The format of all data inputs and provisions for error detection and correction.
  - The format for all status and error messages and data printouts (e.g., formats, headings, data units, abbreviations, spacing, columns).
- Demonstrate any rapid design prototypes used to make design decisions.
- Identify potential high risk areas in the design and any requirements changes that could reduce risk.
- Review to assure that consideration has been given to optimizing the maintainability and maintenance aspects of the software product.

Review Items: The following items should be considered for review and evaluation during the Functional Design Review. Be prepared to discuss in technical detail any of these items within the scope of the review.

- Functional flows. Indicate how the computer software functional flows map the software and interface requirements to the individual high-level components of the software product.

Review Items,

continued:

- Storage allocation data. Describe the manner in which available storage is allocated to individual software components. Timing, sequencing requirements, and relevant equipment constraints used in determining the allocation should be included.

- Control functions. Describe the executive control and start/recovery features of the software product.

- Component structure. Describe the high-level structure of the software product, the reasons for choosing the components, the development methodology that will be used within the constraints of available computer resources, and any support programs that will be required in order to develop and maintain the software product and allocated data storage.

- Security. Identify the security requirements and provide a description of the techniques to be used for implementing and maintaining security within the software product.

- Computer software engineering facilities. Describe the availability, adequacy, and planned utilization of the computer software engineering facilities including both Government-provided and commercially available facilities.

- Computer software engineering facility versus the operational system. Describe any unique design features that exist in the functional design in order to allow use within the computer software engineering facility that will not exist in the operational software product. Provide information on the design of support programs not explicitly required for the operational system that will be generated to assist in the development of the software product.

- Development tools. Describe any special tools (e.g., simulation, data reduction, or utility tools) that are not deliverables, but are planned for use during software development.

- Test tools. Describe any special test systems, test data, data reduction tools, test computer software, or calibration and diagnostic software that are not deliverables, but are planned for use during software development.

Review Items, continued:

- Commercial resources. Describe commercially available computer resources, including any optional capabilities (e.g., special features, interface units, special instructions, controls, formats). Identify any limitations of commercially available equipment (e.g., failure to meet user interface, safety, and maintainability requirements) and identify any deficiencies.

- Existing documentation. Maintain a file and have available for review any existing documentation supporting the use of commercially available computer resources.

- Support resources. Describe the resources necessary to support the software product during engineering, installation, and operational state (e.g., operational and support hardware and software personnel, special skills, human factors, configuration management, testing support, documentation, and facilities/space management).

- Operation and support documentation. Describe the documentation that will be produced to support the operation and maintenance of the software product.

Work Product: Create and distribute official meeting minutes for each session. The minutes should consist of significant questions and answers, action items and individual/group responsible, deviations, conclusions, and recommended courses of action resulting from presentations or discussions. Recommendations that are not accepted should be recorded along with the reason for non-acceptance. Minutes must be distributed to the system owner and users for review and notification of review performance as follows:

- Approval - indicates that the functional design is satisfactorily completed.

- Contingent Approval - indicates that the functional design is not considered accomplished until the satisfactory completion of resultant action items.

- Disapproval - indicates that the functional design is inadequate. Another Functional Design Review is required.

Activity: **5.9**
Initiate Procurement of Hardware and Software

Responsibility: Project Manager/Team

Description: Careful consideration should be given to purchasing off-the-shelf software before expending the time, resources, and costs associated with developing custom-built systems. Whenever possible, acquire off-the-shelf software to satisfy some or all of the project requirements. In addition, some projects may require the acquisition of hardware or software to support the design, code, and test processes.

Try to acquire a demonstration package of any proprietary software before completing the design specifications. The proprietary software may prove inadequate or inappropriate once it has been evaluated through hands-on use. Create a pilot of the software product to exercise the most important functions provided by the proprietary software as well as to obtain definite performance indications.

Initiate the procurement of any hardware or software well in advance of the planned need for these products. Adequate time must be allocated in the Project Plan timeline for the selection, procurement, installation, testing, and training associated with each vendor product.

The project team may assume all of the procurement, installation, and testing responsibilities, or the acquisition and testing of some hardware and software may be initiated by the functional area that is most familiar with the product. For example, a local area network engineering group may procure and test local area network or client/server software; a mainframe systems group may procure and test mainframe software.

Note: When the expected operating platform for a software product will require extensive procurement of hardware and software, it is recommended that procurement needs be addressed as early in the lifecycle as possible. If hardware and software acquisition requirements are known, develop the Acquisition and Installation Plans for all operating sites and initiate the procurement process. Review and, if necessary, revise the Acquisition and Installation Plans at the beginning of the Programming Stage. Requirements for the Acquisition and Installation Plans are provided in *Chapter 7, Programming Stage.*

Work Product: Place a copy of all software and hardware procurement records (e.g., justifications, approvals, purchase orders, and invoices) and the Acquisition and Installation Plans (if developed) in the Project File.

Activity: **5.10**
Revise Project Plan

Responsibility: Project Manager

Description: Once the Functional Design Review is completed and the functional design is baselined, determine if the project estimates for resources, cost, and schedule need to be revised and if the selected design approach is still appropriate for the size and complexity of the project.

Work Product: Review the Project Plan for accuracy and completeness of all Functional Design Stage activities and make any changes needed to update the information. Expand the information for the System Design Stage to reflect accurate estimates of resources, costs, and hours.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Review Process: Conduct a structured walkthrough to assure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the System Design Stage.

The Project Plan is formally reviewed during the In-Stage Assessment and Stage Exit processes.

Activity: **5.11**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the Functional Design Stage Exit process. Additional ISAs can be performed during the stage, as appropriate. The completion of the Functional Design Document is an appropriate time to schedule an ISA.

Provide the reviewer with copies of all work products developed or revised during the Functional Design Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/best practices.
- Sound project management practices are being used.
- Project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide*. A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **5.12**
Conduct Functional Design Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the Functional Design Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the guide is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the System Design Stage.

Software Engineering Methodology

Chapter 6.0
System Design Stage

Table of Contents

Chapter **Page**

6.0 System Design Stage 6.0-1
- 6.1 Select System Architecture 6.1-1
  - 6.1.1 Evaluate System Architecture Alternatives 6.1-2
  - 6.1.2 Recommend System Architecture 6.1-5
- 6.2 Design Specifications for Software Modules 6.2-1
- 6.3 Design Physical Model and Data Base Structure 6.3-1
- 6.4 Develop Integration Test Plan 6.4-1
- 6.5 Develop System Test Plan 6.5-1
- 6.6 Develop Conversion Plan 6.6-1
- 6.7 Develop System Design 6.7-1
  - 6.7.1 Develop System Design Document 6.7-2
  - 6.7.2 Conduct Critical Design Review 6.7-3
- 6.8 Develop Program Specifications 6.8-1
- 6.9 Define Programming Standards 6.9-1
- 6.10 Revise Project Plan 6.10-1
- 6.11 Conduct In-Stage Assessment 6.11-1
- 6.12 Conduct System Design Stage Exit 6.12-1

Chapter: **6.0**
System Design Stage

Description: The goal of this stage is to translate the user-oriented functional design specifications into a set of technical, computer-oriented system design specifications; and to design the data structure and processes to the level of detail necessary to plan and execute the Programming and Installation Stages. General module specifications should be produced to define what each module is to do, but not how the module is to be coded. Effort focuses on specifying individual routines and data structures while holding constant the software structure and interfaces developed in the previous stage. Each module and data structure is considered individually during detailed design with emphasis placed on the description of internal and procedural details. The primary work product of this stage is a software system design that provides a blueprint for the coding of individual modules and programs.

Input: The following items provide input to this stage.

- Project File
- Design records
- Logical model
- Data dictionary *(expanded)*
- Requirements Traceability Matrix *(expanded)*
- Functional Design Document
- Hardware and software procurement records
- Project Plan *(revised)*
- Software Quality Assurance Plan

High-Level Activities: The remainder of this chapter is divided into sections that describe the specific high-level activities performed during this stage. These activities represent the minimum requirements for a large software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

6.1 Select System Architecture
6.2 Design Specifications for Software Modules
6.3 Design Physical Model and Data Base Structure
6.4 Develop Integration Test Plan
6.5 Develop System Test Plan
6.6 Develop Conversion Plan
6.7 Develop System Design

High-Level

Activities,continued:

6.8 Develop Program Specifications
6.9 Define Programming Standards
6.10 Revise Project Plan
6.11 Conduct In-Stage Assessment
6.12 Conduct System Design Stage Exit

Output: Several work products are produced during this stage. The work products listed below are the minimum requirements for a large software project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

- Design specifications
- Physical Model
- Data Dictionary *(expanded)*
- Integration Test Plan *(draft)*
- System Test Plan *(draft)*
- Conversion Plan
- Requirements Traceability Matrix *(expanded)*
- System Design Document
- Program Specifications
- Programming Standards
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 6.0-1, System Design Stage Activities and Work Products by Project Size*. The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large projects.

Review Process: Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. In addition, a Critical Design Review is conducted once the System Design Document is developed. This review is an important milestone in the design process. The time and resources needed to conduct the walkthroughs and Critical Design Review should be indicated in the project resources, schedule, and work breakdown structure.

Reference: *Appendix C, Conducting Structured Walkthroughs,* provides a procedure and sample forms that can be used for structured walkthroughs.

Exhibit 6.0-1. System Design Stage Activities and Work Products by Project Size

Work Activity		Project Size L	Project Size M	Project Size S	Work Product	Scheduled Deliverables L	Scheduled Deliverables M	Scheduled Deliverables S
6.1	Select System Architecture	A	A	A	Analysis of Benefits and Costs Report Summary and recommendations of architecture alternatives	A A	A A	A A
6.2	Design Specifications for Software Modules	R	R	R	Design diagrams with text	R	R	R
6.3	Design Physical Model and Data Base Structure	R	R	R	Data Dictionary *(expanded)* Physical Model	R R	R R	R R
6.4	Develop Integration Test Plan	R	R	R	Integration Test Plan *(draft)*	R	R	R
6.5	Develop System Test Plan	R	R	R	System Test Plan *(draft)*	R	R	R
6.6	Develop Conversion Plan	A	A	A	Conversion Plan	A	A	A
6.7	Develop System Design	R	R	R	Requirements Traceability Matrix *(expanded)* System Design Document Critical Design Review minutes	R R R	R R R	R R R
6.8	Develop Program Specifications	R	R	R	Program Specifications	R	R	R
6.9	Define Programming Standards	R[2]	R[2]	0	Programming Standards	R[2]	R[2]	R[2]
6.1	Revise Project Plan	R	R	R	Project Plan *(revised)*	R	R	R
6.11	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
6.12	Conduct System Design Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
N = Not Applicable

Minimum Requirements: R = Required
A = As Appropriate

[1] = Completed by reviewer
[2] = Can use existing plan/procedure

S= Small

Bibliography: The following materials were used in the preparation of the System Design Stage chapter.

1. Barker, Richard, *CASE*METHOD*, Tasks and Deliverables, 1990. pp. 5-10 and 5-11.

2. Bramucci, Wilma, "Systems Development Software," *Faulkner Technical Reports, Inc.*, June 1989. pp. 1-7.

3. Gane, Chris and Sarson, Trish, *Structured Systems Analysis: Tools and Techniques*, Prentice-Hall, Inc., Englewood Cliffs, 1979.

4. *Software Engineering Handbook* Chapter 5, Software Design; and Chapter 7, Software Testing.

5. *Software System Testing and Quality Assurance*, Chapter 5, Integration.

6. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Requirements Specifications*, ANSI/IEEE Std 830-1984, New York, 1984.

7. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Guide to Software Design Descriptions*, IEEE Std 1016.1-1993, New York, 1993.

8. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Recommended Practice for Software Design Descriptions*, IEEE Std 1016-1987, New York, 1987.

9. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes*, IEEE Std 1074-1991, New York, 1992.

10. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Verification and Validation Plans*, ANSI/IEEE Std 1012-1986, New York, 1986.

11. U.S. Department of Commerce, National Bureau of Standards, *Guideline for Planning and Management of Database Applications*, Federal Information Processing Standards Publication 77, 1980. pp. 10-23.

12. U.S. Department of Defense, Military Standard, *Defense System Software Development*, MIL-STD-2167A, 1988. pp. 27-29.

13. U.S. Department of Defense, Military Standard, *Specification Practices*, MIL-STD-490 B, 1986. pp. 47-50.

14. U.S. Department of Defense, Military Standard, *Technical Reviews and Audits for Systems, Equipments, and Computer Software*, MIL-STD-1521 B, 1985. pp.5, 33-52.

15. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1*, K/CSD/INF/86-3, Vol.1,R3, prepared under contract by Martin Marietta Energy Systems, Inc. at Oak Ridge National Laboratory, August 1987.

16. U.S. Department of Energy, *Hanford Site Data Administration Guide*, RLO-ADP-1, July 1989.

Bibliography, continued:

17. U.S. Department of Energy, *Software Management Guide*, DOE/AD-0028, 1992.

18. U.S. Department of Energy, Nevada Operations Office, *Software Management Plan*, May 1991.

19. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual*, 1988.

20. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual*, 1988.

21. U.S. Social Security Administration, Office of Systems, *Software Engineering Technology (SET) Manual*, Volume 1, 1990. Part 40 Design Stage.

22. Yourdon, inc., *Management Overview of Real-Time Structured Analysis and Design*, Edition 3.0, January 1984.

Activity: **6.1**
Select System Architecture

Responsibility: Project Team

Description: When the system architecture for the software product has not been predetermined by the existing computing environment of the system owner and users, evaluate system architecture alternatives to determine which one has the best, cost-effective solution that satisfies the project requirements.

"Cost effective solution" does not imply the least expensive alternative. The "best, cost effective solution" is the alternative that does the best job of satisfying the project requirements, assures the highest quality software product, and provides for an adequate return on investment in a timeframe that is acceptable to the system owner.

Select the specific hardware, software, data base management system, and communication facilities based on the following types of considerations.

- Departmental or site-specific information architecture guidelines or standards
- Hardware and software that emphasizes simplicity, flexibility, ease of operation and maintenance
- Cost to procure and maintain potential environment
- Backup and recovery procedures
- Selection of a distributed or centralized processing environment
- Communication requirements
- Data configuration

Obtain support from functional area points-of-contact to aid in the architecture evaluation process. Consultations and input may be helpful from system and data base administrators, local area network administrators, operations personnel, system programmers, and telecommunication experts.

The following tasks are involved in selecting a system architecture.

6.1.1 Evaluate System Architecture Alternatives
6.1.2 Recommend System Architecture

Task: **6.1.1**
Evaluate System Architecture Alternatives

Description: Consider system architecture alternatives within the site's information architecture guidelines that enable the project objectives and requirements to be achieved. The selection of a system architecture depends on many factors such as the experience of the project team with each alternative and the availability of reusable components to facilitate the implementation of an alternative.

When investigating alternatives, consider the following issues.

- Those functions or portions of functions that are to be automated and the functions that will be manual. Conduct an examination of *what* the automated portion of the project will encompass.

- The technical solution for the objectives. The determinations of *how* the software product is to be designed; (e.g., online vs. batch, client-server vs. mainframe, Oracle vs. Sybase).

- The system owner's and users' computing environment and the needs created by the technical solution. Consider any hardware and software that must be acquired, including system access software, operating system software, data base management system, and telecommunications facilities.

Procedure: The following procedure provides one approach for evaluating the architecture alternatives.

- Conduct an Analysis of Benefits and Costs to determine the most cost effective alternative. On the benefits side, include the improvements over the current process being used to support the business application. On the costs side, include any degradation from current capabilities along with the rationale for allowing the degradation.

- Create and evaluate a data flow diagram for each alternative.

- Identify how users would interact with the features associated with each alternative (such as the generation of queries and reports).

- Create a list of the risks associated with each alternative and develop a plan for mitigating each risk.

Procedure, continued:

- Compare the performance capabilities of each alternative. How fast will each alternative be able to process the user's work given a particular hardware resource. Performance is usually expressed in terms of throughput, run time, or response time. Five factors that frequently affect performance include:

  - Number of intermediate files in a system (park data between programs)
  - Number of times a given file is passed
  - Number of seeks against a disk file
  - Time spent in calling programs and other system overhead
  - Time taken to execute actual program

- Compare the security and access control features of each alternative. To what extent does the alternative provide security against human errors, machine malfunction, or deliberate mischief. Some common controls include:

  - Check digits on predetermined numbers
  - Batch control totals
  - Creation of journals and audit trails
  - Limited access to files

- Compare the ease with which each alternative allows the system to be modified to meet changing requirements, such as:

  - Fixing errors
  - Changing user needs
  - Mandatory/statutory modifications
  - Enhancements

Work Product: Maintain records on each alternative that is evaluated. Use this information to develop a summary of the system architecture alternatives. The summary will be integrated into the materials presented to the system owner when a system architecture recommendation is made. Place a copy of the records for each alternative and the summary in the Project File.

Work Product continued:

If an Analysis of Benefits and Costs (ABC) is conducted, prepare a report that describes the process used for the analysis, a summary of the alternatives considered, and the results obtained. The report will be integrated into the materials presented to the system owner when a system architecture recommendation is made. Place a copy of the ABC records and report in the Project File.

References:

The following documents provide detailed guidance on conducting an Analysis of Benefits and Costs.

- *Analysis of Benefits and Costs (ABC's) Guideline. Volume 1, A Manager's Guide to Analysis of Benefits and Costs.* U.S. Department of Energy.

- *Analysis of Benefits and Costs (ABC's) Guideline. Volume 2, An Analyst's Handbook for Analysis of Benefits and Costs.* U.S. Department of Energy.

Task: **6.1.2**
Recommend System Architecture

Description: Based on the results of the architecture alternatives evaluation, develop a recommendation for a system architecture that is cost-effective and will facilitate the achievement of the software project requirements. Prepare a presentation for the system owner and users that provides the following types of information to support the recommendation.

- Review the limitations or problems with any current manual or automated system that will be resolved by the software product.
- Present the logical model for the software product. Highlight new functions that would be incorporated.
- For each architecture alternative that was evaluated, present the following information.
  - A description of the alternative.
  - An overall data flow diagram showing how the alternative would be implemented.
  - The way the system would look to the users, in terms of hardware, user interface, reports, and query facilities.
  - The estimated benefits of the alternative.
  - The estimated cost and time to implement the alternative.
  - A statement of the element of risk associated with the alternative.
- Present the recommended alternative and explain why it was selected.

Before the project proceeds, the system owner should make a decision about the system architecture either by formally accepting the project team's recommendation or by directing the team to use a different architecture. Any delay in making this decision could result in a slippage of the project schedule.

Work Product: Document the project team's recommendation for the most cost-effective and viable architecture alternative. Provide a summary of each alternative that was evaluated. Describe the rationale for proposing the recommended architecture. Describe the impact of this alternative on the system owner and users organization(s) and other systems. Include any background information that was relevant to the decision process, such as the Analysis of Benefits and Costs Report.

Present the project team's recommendation for the system architecture to the system owner and users. The recommendation can be delivered as a document or as a presentation. Place a copy of the document or presentation materials in the Project File.

Review Process: Conduct a structured walkthrough to assure that the most cost-effective and viable architecture alternative is being recommended.

Activity: **6.2**
Design Specifications for Software Modules

Responsibility: Project Team

Description: During the Functional Design Stage, a decomposition of the software product requirements resulted in a collection of design entities (or objects). In the System Design Stage, these design entities are grouped into the routines, modules, and programs that need to be developed or acquired as off-the-shelf or reusable software.

Expand the functional design to account for each major software action that must be performed and each data object to be managed. Detail the design to a level such that each program represents a function that a programmer will be able to code.

Procedure: Use the following procedure to design the software module specifications.

- Identify a software program for each action needed to meet each function or data requirement in the Software Requirements Specification and the data dictionary.

- Identify any routines and programs that may be available as reusable code or objects from existing applications or off-the-shelf software. The System Review Inventory System (SRIS) maintained at DOE Headquarters and the Energy Science and Technology Software Center (ESTSC) located at Oak Ridge, Tennessee are recommended sources for identifying reusable software. The ESTSC is the Department's central collection of DOE-supported software packages. The Center also collects software from the Nuclear Regulatory commission and others, and maintains contact with other software centers.

- Identify programs that must be designed and developed (custom-built). Assign a name to each program and object that is functionally meaningful. Identify the system features that will be supported by each program.

- Specify each program interface. Update the data dictionary to reflect all program and object interfaces changed while evolving from the functional to the system design.

Procedure, continued:

- Define and design significant attributes of the programs to be custom-built.
- Expand the program interfaces to include control items needed for design validity (e.g., error and status indicators).
- Combine similar programs and objects. Group the design entities into modules based on closely knit functional relationships. Formulate identification labels for these modules.
- Show dependencies between programs and physical data structures (e.g., files and global tables). Avoid defining a program that not only needs data residing in a file or global table, but also depends on the physical structure or location of data.
- Change the design to eliminate features that reduce maintainability and reusability (i.e., minimize coupling between programs and maximize the cohesion of programs).

Work Product: Document the system design primarily in the form of diagrams. Supplement each diagram with text that summarizes the function (or data) and highlights important performance and design issues.

When using structured design methods, the design diagrams should:

- Depict the software as a top-down set of diagrams showing the control hierarchy of all software programs to be implemented.
- Define the function of each software program.
- Identify data and control interfaces between programs.
- Specify files, records, and global data accessed by each program.

When using object-oriented or data-centered design methods, the design diagrams should:

- Show the data objects to be managed by the software.

Work Product, continued:

- Specify the program functions to be included within each object.
- Identify functional interfaces between objects.
- Specify files and records comprising each object.
- Identify relationships between data files.

Review Process: Conduct structured walkthroughs to assure that the custom-built routines and programs are correctly designed.

Activity: **6.3**
Design Physical Model and Data Base Structure

Responsibility: Project Team

Description: The physical model is a description of the dynamics, data transformation, and data storage requirements of the software product. The physical model maps the logical model created during the Functional Design Stage to a specific technical reality. Care must be taken to retain in the physical implementation all of the capabilities inherent in the logical model.

The physical model frequently differs from the logical model in the following areas.

- Constraints imposed by the data base management system - The logical data model may have different implementations in the selected data base management system.

- Performance - Data redundancies, indices, and data structure changes may have to be introduced into the physical model to improve performance.

- Distributed processing - Possible network and multiple production hardware configurations may cause changes to the physical data model.

Designing the data base structure converts the data requirements into a description of the master and transient files needed to implement the requirements. If the software product will include a data base, design the data base in conjunction with the following data base management features.

- Report writer and file processing capabilities
- Online query processing to retrieve data
- Automated data dictionary systems

Work Product: Document the physical model for incorporation into the System Design Document. Review the contents of the data dictionary entries and update to complete information on data elements, entities, files, physical characteristics, and data conversion requirements. Place a copy of all physical model and data base structure records in the Project File.

Review Process: Schedule structured walkthroughs to verify that the physical model and data dictionary are correct and complete.

Activity: **6.4**
Develop Integration Test Plan

Responsibility: Project Team Programmers

Description: The purpose of integration testing is to verify the integrity of a module (a cohesive set of programs) and its interfaces with other modules within the software structure. An integration test plan is developed to incorporate successfully unit-tested modules into the overall software structure and to test each level of integration to isolate errors introduced by newly incorporated modules.

The number of integration levels, the classes of tests to be performed, and the order in which routines and builds are incorporated into the overall software structure are addressed in the Integration Test Plan. The following factors should be considered.

- Are routines to be integrated in a pure top-down manner or should builds be developed to test subfunctions first?
- In what order should major software functions be incorporated?
- Is the scheduling of module coding and testing consistent with the order of integration?
- Is special hardware required to test certain routines?

Integration testing should include tests that validate the following functions.

- Verify each interface between the module and all other modules.
- Access each input message or command processed by the module.
- Check each external file or data record referenced by coding statements in the module.
- Output each message, display, or record generated by the module.

An important consideration during integration test planning is the amount of test software (e.g., drivers, test case generation) that must be developed to adequately test the required functionality.

Description, continued:

For example, it may be cost-effective to delay testing of a communication function until hardware is available rather than generate test software to simulate communication links. Similarly, it may be better to include certain completed modules in the software structure in order to avoid having to develop software drivers. These decisions are made on the basis of cost and risks.

Work Product:

Develop the draft Integration Test Plan that addresses the following activities.

- Define the integration tests at each element level, stating objectives, what is to be tested, and verified. Testing is from the point of view of structure and function.
- Define all aspects of the formal interfaces that must undergo formal integration testing. Review interface requirements to ensure completeness, consistency, and effectiveness.
- Plan for test tools and software that must be developed to adequately test the required functionality.

Review Process:

Conduct a peer review or structured walkthrough to assure that the draft Integration Test Plan is accurate and complete. The Integration Test Plan will be reviewed and revised as needed during the Programming Stage.

Activity: **6.5**
Develop System Test Plan

Responsibility: Project Test Team

Description: The objectives of the system test process are to assure that the software product adequately satisfies the project requirements; functions in the computer operating environment; successfully interfaces between user procedures, operating procedures, and other systems; and protects the software and data from security risks. The system should be tested under the same kind of daily conditions that will be encountered during regular operations. System timing, memory, performance, and security functions are tested to verify that they perform as specified. The functional accuracy of logic and numerical calculations are tested for verification under normal and load conditions.

Test data should be varied and extensive enough to enable the verification of the operational requirements. Expected output results should be included in the test plan in the form of calculated results, screen formats, hardcopy output, predetermined procedural results, warnings, error messages and recovery.

Detailed planning for the system testing helps to ensure that system acceptance will be successfully completed on schedule. When applicable, system testing must include the following types of tests.

- Performance tests that measure throughput, accuracy, responsiveness, and utilization under normal conditions and at the specified maximum workload.
- Stress tests to determine the loads that result in appropriate, non-recoverable, or awkward system behavior.
- Interface tests to verify that the system generates external outputs and responds to external inputs as prescribed by approved interface control documentation.
- System recovery and reconfiguration tests.
- Verification that the system can be properly used and operated in accord with its users guide and operating instructions.

Description, continued:

- Verification that the system meets its requirements for reliability, maintainability, and availability, including fault tolerance and error recovery.

- Verification of the effectiveness of error detection and analysis, and automated diagnostic tools.

- Demonstration that the system complies with its serviceability requirements such as accessibility, logistics, upgrades, diagnostics, and repair capabilities.

Work Product: Develop a draft System Test Plan that describes the testing effort, provides the testing schedule, and defines the complete range of test cases that will be used to assure the reliability of the software. The test cases must be complete and the expected output known before testing is started. The test plan should address the following.

- Provide a definition of, and the objectives for, each test case.

- Define the test scenario(s) including the step-by-step procedure, the number of processing cycles to be tested or simulated, and the method and responsibility for feeding test data to the system.

- Define the test environment including the hardware and software environment under which the testing will be conducted. Identify and describe manual procedures, automated procedures, and test sites (real or simulated).

- Identify test tools and special test support needs (e.g., hardware and software to simulate operational conditions or test data that are recordings of live data).

- Identify responsibilities for conducting tests; for reviewing, reporting, and approving the results; and for correcting error conditions.

- Develop a requirements verification matrix mapping individual tests to specific requirements and specifying how each system requirement will be validated.

- Schedule for integrating and testing all components including adequate time for retesting.

Review Process: Conduct peer reviews or structured walkthroughs to assure that each system test procedure is accurate, complete, and accomplishes the stated objectives. The System Test Plan will be reviewed and revised as needed during the Programming Stage.

Activity: **6.6**
Develop Conversion Plan

Responsibility: Project Team

Description: If the software product will replace an existing automated system, develop a Conversion Plan. The major elements of the Conversion Plan are to develop conversion procedures, outline the installation of new and converted files/data bases, coordinate the development of file-conversion programming, and plan the implementation of the conversion procedures.

File conversion should include a confirmation of file integrity. Determine what the output in the new system should be compared with the current system, and ensure that the files are synchronized. The objective of file conversion is new files that are complete, accurate and ready to use.

Many factors influence data conversion, such as the design of the current and new systems and the processes for data input, storage, and output. Understanding the data's function in the old system and determining if the function will be the same or different in the new system is of major importance to the Conversion Plan. The structure of the data to be converted can limit the development of the system and affect the choice of software.

Work Product: Develop a Conversion Plan that identifies what conversions are needed and how the conversion(s) will be implemented. Consider the following factors during the development of the conversion Plan.

- Determine if any portion of the conversion process should be performed manually.
- Determine whether parallel runs of the old and new systems will be necessary during the conversion process.
- Understanding the function of the data in the old system and determining if the use will be the same or different in the new system is important.
- The order that data is processed in the two systems influences the conversion process.

Work Product, continued:

- Volume considerations, such as the size of the data base and the amount of data to be converted, influence how the data will be converted. Especially important are the number of reads that are necessary, and the time these conversions will take.

- User work and delivery schedules, timeframes for reports and end-of-year procedures, and the criticality of the data help determine when data conversion should be scheduled.

- Determine whether data availability and use should be limited during the conversion.

- Plan for the disposition of obsolete or unused data that is not converted.

Review Process: Conduct structured walkthroughs to assure that the Conversion Plan is accurate and complete.

Activity: **6.7**
Develop System Design

Responsibility: Project Team

Description: The system design is the main technical work product of the System Design Stage. The system design translates requirements into precise descriptions of the software components, interfaces, and data necessary before coding and testing can begin. It is a blueprint for the Programming Stage, based on the software structure and data model established in the Functional Design Stage.

The system design plays a pivotal role in the development and maintenance of a software product. The design provides valuable information used by the project manager, quality assurance staff, configuration management staff, software designers, programmers, testers, and maintenance personnel.

The system design is baselined after the system owner's formal approval of the design as described in the System Design Document. Once the system design is baselined, any changes to the design must be managed under change control procedures established in the Software Configuration Management Plan. Approved changes must be incorporated into the System Design Document.

It is important for the system owner/users to understand that some changes to the baselined system design may affect the project scope and therefore can change the project cost, resources, or schedule. It is the responsibility of the project manager and team to identify system owner/user requested changes that would result in a change of project scope; evaluate the potential impact to the project costs, resources, or schedule; and notify the system owner of the project planning revisions that will be required to accommodate their change requests.

Work Product: Each requirement identified in the Software Requirements Specification must be traceable to one or more design entities. This traceability ensures that the software product will satisfy all of the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage to relate the system design to the requirements. Place a copy of the expanded matrix in the Project File.

The following tasks are involved in developing the system design.

6.7.1 Develop System Design Document
6.7.2 Conduct Critical Design Review

Task: **6.7.1**
Develop System Design Document

Description: The System Design Document records the results of the system design process and describes how the software product will be structured to satisfy the requirements identified in the Software Requirements Specification. The System Design Document is a translation of the requirements into a description of the software structure, software components, interfaces, and data necessary to support the programming process.

Work Product: Prepare the System Design Document and submit it to the system owner and users for their review and approval. The approved System Design Document is the official agreement and authorization to use the design to build the software product. Approval implies that the design is understood, complete, accurate, and ready to be used as the basis for the subsequent lifecycle stages. Place a copy of the approved System Design Document in the Project File.

Review Process: Conduct structured walkthroughs as needed to ensure that the System Design Document is accurate and complete.

The completion of the System Design Document is an appropriate time to schedule an In-Stage Assessment (ISA). The *In-Stage Assessment Process Guide* provides a description and instructions for conducting an ISA. A copy of the guide is provided in Appendix D.

Task: **6.7.2**
Conduct Critical Design Review

Description: The Critical Design Review is a formal technical review of the system design. The purpose of the review is to demonstrate to the system owner and users that the system design can be implemented on the selected platform and accounts for all software and data requirements and accommodates all design constraints (e.g., performance, interface, security, safety, resource, and reliability requirements). The design review should include a review of the validity of algorithms needed to perform critical functions.

Several short Critical Design Reviews can replace one long review if the software consists of several components that are not highly interdependent. The review process should be a series of presentations by the project team to the system owner and other approval authorities.

Conduct a Critical Design Review that demonstrates that the design specifications are capable of supporting the full functionality of the software product, as follows:

- All algorithms will perform the required functions.
- The specification is complete, unambiguous and well documented, including timing and sizing, and data and storage allocations.
- The specification is necessary and sufficient for, and directly traceable to, the software system design.
- The specification is compatible with every other specification, piece of equipment, facility, and item of system architecture, especially as regards information flow, control, and sequencing.
- The specification is consistent with the abilities of current development and user personnel.

In addition to verifying individual specifications, the Critical Design Review assesses other project work products to ensure the following.

- The approved design approach is being followed by the team.
- Measures to reduce risk on a technical, cost, and schedule basis are adequate.

Description, continued:

- The performance characteristics of the design solution are acceptable.
- Testing will be sufficient to ensure software product correctness.
- The resultant application will be maintainable.
- Provisions for automatic, semi-automatic, and manual recovery from hardware/software failures and malfunctions are adequate and documented.
- Diagnostic programs, support equipment, and commercial manuals all comply with the system maintenance concept and specification requirements.

Work Product: Create and distribute official meeting minutes for each design review session. The minutes should consist of significant questions and answers, action items and individual/group responsible, deviations, conclusions, and recommended courses of action resulting from presentations or discussions. Recommendations that are not accepted should be recorded along with the reason for non-acceptance. Minutes must be distributed to review participants. The system owner determines review performance as follows:

- Approval - The review was satisfactorily completed.
- Contingent Approval - The review is not finished until the satisfactory completion of resultant action items.
- Disapproval - The specification is inadequate. Another Critical Design Review will be required.

Activity: **6.8**
Develop Program Specifications

Responsibility: Project Team

Description: A Program Specification is a written procedural description of each software system routine. The Program Specification should provide precise information needed by the programmers to develop the code.

Many techniques are available for specifying the system design, such as formal specification languages, program design languages (e.g, pseudo-code or structured English), meta-code, tabular tools (e.g., decision tables), and graphical methods (e.g., flow charts or box diagrams). In object-oriented design, the specification of requirements and preliminary design constraints and dependencies often results in the design language producing the detailed specifications.

Select the technique or combination of techniques that is best suited to the software project and to the experience and needs of the programmers who will use the system design as their blueprint. The following are suggestions for using the techniques.

- Decision trees are useful for logic verification or moderately complex decisions that result in up to 10-15 actions. Decision trees are also useful for presenting the logic of a decision table to users.

- Decision tables are best used for problems involving complex combinations of up to 5-6 conditions. Decision tables can handle any number of actions; however, large numbers of combinations of conditions can make decision tables unwieldy.

- Structured English is best used wherever the problem involves combining sequences of actions with decisions or loops. Once the main work of physical design has been done and physical files have been defined, it becomes extremely convenient to be able to specify physical program logic using the conventions of structured English, but without getting into the detailed syntax of any particular programming language (pseudo-code).

- Standard English is best used for presenting moderately complex logic once the analyst is sure that no ambiguities can arise.

Work Product: Specifications may be produced as documents, graphic representations, formal design languages, records in a data base management system, and CASE tool dictionaries. A list of significant program attributes typically included in a Program Specification is provided at the end of this section.

Review Process: Conduct a series of structured walkthroughs to ensure that the Program Specification is accurate and complete.

Sample Attributes: For each program to be custom-built, define the program's functional and technical attributes as they become known. The following is a sample list of program attributes.

- Program identification
- Program name
- Program generic type
- Functional narrative
- Program hierarchical features diagram
- Development dependencies and schedule
- Operating environment
  - equipment
  - programming language and version
  - preprocessor
  - operating system
  - storage restrictions
  - security
- Frequency of run
- Data volumes
- Program termination messages
  - normal termination
  - abnormal termination
- Console/printer messages
- Recovery/restart procedures
- Software objectives
- Program input/output diagram
- Data bank information
- Called and calling programs/modules
- Program logic diagrams
- Significant "how-to" instructions
- Telecommunications information

Activity: **6.9**
Define Programming Standards

Responsibility: Project Team Programmers

Description: Programming standards are necessary to ensure that custom-built software has acceptable design and structural properties. Programming standards must be practical, easy to implement, and accepted by the project team. The project team programmers should be the primary developers of the standard. Use a structured approach to programming to allow for easy modification and to facilitate testing and debugging.

The following guidelines are generally applicable to any programming language. Use these guidelines as the basis for the programming standard and add project-specific standards relating to the programming language and tools.

- Control Flow Constructs
  - sequence
  - if-then-else
  - case statement
  - do-while (pretest loop)
  - do-until (post-test loop)
- Module Size
  - Number of executable lines of source code should average 100 lines per unit.
  - Units should contain no more than 200 lines of executable source code.
- Module Design
  - Units do not share temporary storage locations for variables
  - Units perform a single function
  - Avoid self-modifying code
  - Each unit is uniquely named
  - Each unit has a standard format:
    - prologue
    - variable declarations
    - executable statements/comments
  - Use single entry/exit points except for error paths
  - Set comments off from the source code in a uniform manner

Description,

continued:

- Symbolic Parameters
  - Use instead of specific numerics
  - Use for constants, size of data structures, relative position in list
- Naming Conventions
  - Use uniform naming throughout each unit and module to be put under configuration control
  - Use meaningful variable names
  - Do not use keywords as identifiers
- Mixed Mode Operations
  - Avoid mixed mode expressions
  - Add comments in code whenever used
- Error and Diagnostic Messages
  - Design messages to be self-explanatory and uniform
  - Do not require user to perform table lookups
- Style
  - Use conventions such as indentation, white space, and blank lines to enhance readability
  - Align compound statements
  - Avoid "goto" statements.
  - Avoid compound, negative Boolean expressions
  - Avoid nesting constructs beyond five levels deep
  - Avoid deeply nested "if" statements.
  - Use parentheses to avoid ambiguity
  - Include only one executable statement per line
  - Avoid slick programming tricks that may create or encourage defects or be difficult to maintain; the most direct solution is best

Work Product: Create a programming standards document and distribute the document to all project team members. An existing programming standard can be used if it is applicable to the programming language and tools being used for the project.

Review Process: Conduct a peer review to assure that the programming standards are complete and appropriate for the project's programming language and tools.

Activity: **6.10**
Revise Project Plan

Responsibility: Project Manager

Description: Once the Critical Design Review is completed, the system design is baselined, and the work products from the Functional Design Stage have been updated as needed to reflect changes caused by the system design, determine if the project estimates for resources, cost, and schedule need to be revised.

Work Product: Review the Project Plan for accuracy and completeness of all System Design Stage activities and make any changes needed to update the information. Expand the information for the Programming Stage to reflect accurate estimates of resources, costs, and hours. Place a copy of the revised Project Plan in the Project File.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Review Process: Conduct a structured walkthrough to ensure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the Programming Stage.

The Project Plan is formally reviewed during the In-Stage Assessment and Stage Exit processes.

Activity: **6.11**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the System Design Stage Exit process. Additional ISAs can be performed during the stage, as appropriate. The completion of System Design Document is an appropriate time to schedule an ISA.

Provide the reviewer with copies of all work products developed or revised during the System Design Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/best practices.
- Sound project management practices are being used.
- The project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide*. A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **6.12**
Conduct System Design Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the System Design Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the guide is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the Programming Stage.

Software Engineering Methodology

Chapter 7.0
Programming Stage

Table of Contents

Chapter **Page**

7.0 Programming Stage 7.0-1
- 7.1 Develop Acquisition Plan 7.1-1
- 7.2 Develop Installation Plan 7.2-1
- 7.3 Establish Programming Environment 7.3.1
- 7.4 Write Programs 7.4-1
- 7.5 Conduct Unit Testing 7.5-1
- 7.6 Establish Development Baselines 7.6-1
- 7.7 Plan Transition to Operational Status 7.7-1
- 7.8 Generate Operating Documentation 7.8-1
  - 7.8.1 Produce Users Manual 7.8-3
  - 7.8.2 Produce Programmers Reference Manual 7.8-5
- 7.9 Develop Training Program 7.9-1
- 7.10 Revise Project Plan 7.10-1
- 7.11 Conduct In-Stage Assessment 7.11-1
- 7.12 Conduct Programming Stage Exit 7.12-1

Chapter: **7.0**
Programming Stage

Description: In this stage any hardware or software procured to support the programming effort is installed. Plans are developed for the acquisition and installation of the operating environment hardware and software. A training program is designed and a Training Plan that describes the program is produced.

The activities in this stage result in the transformation of the system design into the first complete representation of the software product. The source code, including suitable comments, is generated using the approved program specifications. If the software product requires a data base, any data base utilities are coded. The source code is then grouped into processable units and all high-level language units are compiled into object code. Unit testing is performed to determine if the code satisfies the program specifications and is complete, logical, and error free.

The operating documentation is also produced. The operating documentation is required for installing, operating, and supporting the software product through its lifecycle.

Input: The following items provide input to this stage.

- Project File
- Design specifications
- Physical model
- Data Dictionary
- Integration Test Plan *(draft)*
- System Test Plan *(draft)*
- Conversion Plan
- Requirements Traceability Matrix *(expanded)*
- System Design Document
- Program Specifications
- Programming Standards
- Project Plan *(revised)*
- Software Quality Assurance Plan

High-Level Activities: The remainder of this chapter is divided into sections that describe the specific high-level activities performed during this stage. These activities represent the minimum requirements for a large software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage

High-Level Activities, continued:

requirements to accommodate the different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

7.1 Develop Acquisition Plan
7.2 Develop Installation Plan
7.3 Establish Programming Environment
7.4 Write Programs
7.5 Conduct Unit Testing
7.6 Establish Development Baselines
7.7 Plan Transition to Operational Status
7.8 Generate Operating Documentation
7.9 Develop Training Program
7.10 Revise Project Plan
7.11 Conduct In-Stage Assessment
7.12 Conduct Programming Stage Exit

Output:

Several work products are produced during this stage. The work products listed below are the minimum requirements for a large software project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

- Acquisition Plan
- Installation Plan *(draft)*
- Software units and modules
- Requirements Traceability Matrix *(expanded)*
- Integration Test Plan *(final)*
- System Test Plan *(final)*
- Project Test File
- Development baselines
- Transition Plan
- Operating Documentation *(draft)*
  - Users Manual
  - Programmers Reference Manual
- Training Plan *(draft)*
- Project Plan *(revised)*

Output, continued: A matrix showing the work products associated with each high-level activity is provided in *Exhibit 7.0-1, Programming Stage Activities and Work Products by Project Size*. The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large projects.

Review Process: Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. The time and resources needed to conduct the walkthroughs should be indicated in the project resources, schedule, and work breakdown structure.

References: *Appendix C, Conducting Structured Walkthroughs,* provides a procedure and sample forms that can be used for structured walkthroughs.

Exhibit 7.0-1. Programming Stage Activities and Work Products by Project Size

Work Activity		Project Size			Work Product	Scheduled Deliverables		
		L	M	S		L	M	S
7.1	Develop Acquisition Plan	R	R	A	Acquisition Plan	R	R	A
7.2	Develop Installation Plan	R	R	A	Installation Plan *(draft)*	R	R	A
7.3	Establish Programming Environment	R	R	R		N	N	N
7.4	Write Programs	R	R	R	Completed units and modules of code	I	I	I
7.5	Conduct Unit Testing	R	R	R	Unit test materials Integration Test Plan *(final)* System Test Plan *(final)* Project Test File	I R R R	I R R R	I R R R
7.6	Establish Development Baselines	R	R	R	Baselined code	I	I	I
7.7	Plan Transition to Operational Status	R	R	R	Transition Plan	R	R	R
7.8	Generate Operating Documentation	R	R	R	Users Manual *(draft)* Programmers Reference Manual *(draft)*	R R	R R	R R
7.9	Develop Training Program	R	R	A	Training Plan *(draft)*	R	R	A
7.1	Revise Project Plan	R	R	A	Project Plan *(revised)*	R	R	A
7.11	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
7.12	Conduct Programming Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
S = Small

Minimum Requirements: R = Required
A = As Appropriate
N = Not Applicable

I = Input to other deliverables
[1] = Completed by reviewer
[2] = Can use existing plan/procedure

Bibliography: The following materials were used in the preparation of the Programming Stage chapter.

1. *Software Engineering Handbook*, Chapter 7, Software Testing.

2. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes*, IEEE Std 1074-1991, New York, 1992.

3. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software User Documentation*, IEEE Std 1063-1987, New York, 1988.

4. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1*, K/CSD/INF/86-3, Vol.1,R3, prepared under contract by Martin Marietta Energy Systems, Inc. at Oak Ridge National Laboratory, August 1987.

5. U.S. Department of Energy, *DOE/NV Software Management Plan*, Nevada Operations Office, May 1991.

6. U.S. Department of Justice, Immigration and Naturalization Service, *System Development Life Cycle Standards Manual*, 1991.

7. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual*, 1988.

8. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual*, 1988.

Activity: **7.1**
Develop Acquisition Plan

Responsibility: Project Team

Description: Develop a plan for the acquisition of any hardware, software, and telecommunications equipment needed to install and operate the software product at all system owner and user sites. The plan should address any special procurements necessary to accommodate the hardware and telecommunications equipment that may exist at a particular site. Acquisition planning must include sufficient lead time to accomplish all procurement, delivery, testing, and installation processes.

It may be necessary to perform a risk analysis of the impact of certain resources not being available when needed. Develop a contingency plan for dealing with needed resources that are acquired later than expected.

Work Product: Work closely with the system owner and representatives from the user sites to assure that all site-specific hardware, software, and telecommunications needs are addressed in the Acquisition Plan.

Place a copy of the Acquisition Plan in the Project File.

Note: For projects that do not require extensive procurement and installation of hardware and software, the Acquisition and Installation Plans can be combined into one work product.

Review Process: Conduct a structured walkthrough to assure that the Acquisition Plan is accurate and complete.

Activity: **7.2**
Develop Installation Plan

Responsibility: Project Team

Description: The Installation Plan is prepared to specify the requirements and procedures for the full-scale installation of the developed software product at the system owner's and all users' work sites. The plan also addresses the installation of any hardware, off-the-shelf software, firmware, and telecommunications equipment needed to operate the product at each site. In developing an Installation Plan consider each site's requirements for continuity of operations, level of service, and the needs of the project team, users, maintenance personnel, and management.

Work Product: Work closely with the system owner and representatives from the user sites to assure that all site-specific hardware, software, and telecommunications installation requirements are addressed in the Installation Plan. Develop a draft Installation Plan that addresses the following issues.

- Schedule of all installation activities.
- Items to be delivered to each installation site.
- Number and qualifications of personnel performing installation.
- Equipment environmental needs and installation instructions.
- Hardware, software, firmware, tools, documentation, and space required for each installation.
- Special requirements governing the movement of equipment to each site.
- Telecommunications requirements.
- Dependencies among activities affected by installation.
- Installation tests to assure the integrity and quality of the installed product.

Place a copy of the draft Installation Plan in the Project File.

Note: For projects with limited procurement and installation requirements, the Acquisition and Installation Plans can be combined into one work product.

Review Process: Conduct a structured walkthrough to assure that the draft Installation Plan is accurate and complete. The Installation Plan will be reviewed and revised as needed during the Software Integration and Testing Stage.

Activity: **7.3**
Establish Programming Environment

Responsibility: Project Team

Description: Establishing the programming environment involves assembling and installing the hardware, software, telecommunications equipment, data bases, and other items required to support the programming effort. When the installation of the equipment or software is complete, conduct testing to verify the operating characteristics and functionality of the hardware and software. If required, security software and procedures should be activated when the installations are completed.

If the operational environment is also the programming environment, it may be necessary to alter the operational environment to accommodate an infrastructure of purchased hardware and software for use during programming and testing.

Before being integrated into, or used to support, the software product, vendor products should be tested to verify that the product satisfies the following objectives.

- The product performs as advertised/specified.
- The product's performance is acceptable and predictable in the target environment (e.g., testing for LAN certification).
- The product fully or partially satisfies the project requirements.
- The product is compatible with the project team's other hardware and software tools.

Time should be planned for the project team to become familiar with new products. Ensure that the project team members who will use the hardware or software obtain proper training. This may involve attendance at formal training sessions conducted by the vendor or the services of a consultant to provide in-house training.

This is a good time to review the programming standards that were established in the System Design Stage. Make any changes to the standards that are needed to accommodate the procured hardware and software.

Activity: **7.4**
Write Programs

Responsibility: Project Team Programmers

Description: This activity involves generating the source and object code for the software product. The code should be written in accordance with the programming standards developed in the System Design Stage. Regardless of the platform, development of the code should adhere to a consistent set of programming techniques and error prevention procedures. This will promote reliable, maintainable code, developed in the most efficient and cost effective manner.

The source and object code should be uniquely identified and stored in a way to facilitate the configuration control measures described in the Software Configuration Management Plan.

Writing programs includes the following tasks.

- Use the Program Specifications developed in the System Design Stage as the basis for the coding effort.
- Generate source code and machine-readable modules.
- Generate the physical files and data base structure.
- Generate video screens, report generation codes, and plotting instructions.
- If conversion of an existing system or data is necessary, generate the program(s) described in the Conversion Plan.
- Conduct preliminary testing of completed units. When the test output is correct, review the program specification to assure that the unit or module conforms to the specification.

Coding Practices: The following coding practices should be implemented.

- The programming staff should meet at scheduled intervals to discuss problems encountered and to facilitate program integration and uniformity.

Coding Practices, continued:

- Program uniformity should be achieved by using a standardized set of naming conventions for programs, data elements, variables, and files.
- Modules that can be shared by programs requiring the same functionality should be implemented to facilitate development and maintenance.
- Meaningful internal documentation should be included in each program.
- All code should be backed up on a daily basis and stored in an offsite location to avoid catastrophic loss.
- A standard format for storing and reporting elements representing numeric data, dates, times, and information shared by programs should be determined.
- The System Design Document should be updated to reflect any required deviations from the documented design.

Work Products: The following work products are produced.

- Completed units and modules of code.
- Test materials generated from preliminary testing.

Review Process: Weekly informal reviews of each programmer's work are recommended to keep the project team informed of progress and to facilitate the resolution of any problems that may occur. The combined knowledge and skills of the team members will help to build quality into the software product and support the early detection of errors in design, logic, or code.

Conduct structured walkthroughs on completed units and modules to assure that the code is accurate, logical, internally well documented, complete, and error free. Structured walkthroughs should also be used to validate that the code is reliable and satisfies the program specifications and project requirements.

For large or complex projects, conduct code inspections at successive stages of code production. These inspections are particularly important when code is being produced by several programmers or different programming teams. The inspection team may include experts outside of the project. Ideal times for code inspections occur when code and unit tests are complete, and when the first integration tests are complete. Code inspections should be identified as milestones in the Project Plan.

Activity: **7.5**
Conduct Unit Testing

Responsibility: Project Team Programmers

Description: Unit testing is used to verify the input and output for each module. Successful testing indicates the validity of the function or subfunction performed by the module and shows traceability to the design. During unit testing, each module is tested individually and the module interface is verified for consistency with the design specification. All important processing paths through the module are tested for expected results. All error handling paths are also tested.

Unit testing is driven by test cases and test data that are designed to verify software requirements, and to exercise all program functions, edits, in-bound and out-of-bound values, and error conditions identified in the program specifications. If timing is an important characteristic of the module, tests should be generated that measure time critical paths in average and worst-case situations.

Plan and document the inputs and expected outputs for all test cases in advance of the tests. Log all test results. Analyze and correct all errors and retest the unit using the scenarios defined in the test cases. Repeat testing until all errors have been corrected. While unit testing is generally considered the responsibility of the programmer, the project manager or lead programmer should be aware of the unit test results.

Work Products: Completion of unit testing for a software component signifies internal project delivery of a component or module for integration with other components. Place all components that have completed unit testing under configuration control as described in the Software Configuration Management Plan. Configuration controls restrict changes to tested and approved software.

Review the draft versions of the Integration and System Test Plans developed during the System Design Stage. Update the plans, as needed, to reflect any changes made to the software design. Deliver the final versions of the Integration and System Test Plans to the system owner and user for review and approval. Place a copy of the approved plans in the Project File.

Create a Project Test File for all test materials generated throughout the project lifecycle. Place all unit test materials (e.g., inputs, outputs, results and error logs) in the Project Test File. The test cases used for unit testing may become a subset of tests for integration testing.

Activity: **7.6**
Establish Development Baselines

Responsibility: Project Team Programmers

Description: A development baseline is an approved "build" of the software product. A build can be a single component or a combination of software components. The first development baseline is established after the first build is completed, tested, and approved by the project manager or lead programmer. Subsequent versions of a development baseline should also be approved. The approved development baseline for one build supersedes that for its predecessor build.

Conduct internal build tests such as regression, functional, and performance/ reliability. Regression tests are designed to verify that capabilities in earlier builds continue to work correctly in subsequent builds. Functional tests focus on verifying that the build meets its functional and data requirements and correctly generates each expected display and report. Performance and reliability tests are used to identify the performance and reliability thresholds of each build.

Once the first development baseline is established, any changes to the baseline must be managed under the change control procedures described in the Software Configuration Management Plan. Approved changes to a development baseline must be incorporated into the next build of the software product and revisions made to the affected work products (e.g., Software Requirements Specification, System Design Document, and Program Specifications).

Work Product: Document the internal build test procedures and results. Identify errors and describe the corrective action that was taken. Place a copy of the internal build test materials in the Project Test file.

Maintain configuration control logs and records as required in the Software Configuration Management Plan.

Expand the Requirements Traceability Matrix developed in the Requirements Definition Stage. All work products developed during the code, unit testing, and build processes must be traced back to the project requirements and system design. This traceability ensures that the product will satisfy all of the requirements and remain within the project scope. Place a copy of the expanded Requirements Traceability Matrix in the Project File.

Activity: **7.7**
Plan Transition to Operational Status

Responsibility: Project Team

Description: Successful transition from acceptance testing to full operational use of the software product depends on planning the transition long before the software product is installed in its operational environment. In planning for the transition, quantify the operational needs associated with the software product and describe the procedures that will be used to perform the transition. Rely on experience and data gathered from previous, similar projects to define these needs.

Work Product: Develop a Transition Plan that describes the detailed plans, procedures, and schedules that will guide the transition process. Coordinate development of the plan with the operational and maintenance personnel. The following issues should be considered in the preparation of a Transition Plan.

- Develop detailed operational scenarios to describe the functions to be performed by the operational support staff, maintenance staff, and users.
- Define the number and qualifications of operations and maintenance personnel and specify when they must be in place. Estimate training requirements for these people.
- Document the release process. If development is incremental, define the particular process, schedule, and acceptance criteria for each release.
- Describe the development or migration of data, including the transfer or reconstruction of historic data. Schedule ample time for the system owner and user to review the content of reconstructed or migrated data files to reduce the chance of errors or omissions.
- Specify problem identification and resolution procedures for the operational software product.
- Define the configuration management procedures that will be used for the operational software product. Ideally, the methods defined in the Software Configuration Management Plan that were employed during product development can continue to be used for the operational product.

Work Product, continued:

- Define the scope and nature of support that will be provided by the project team during the transition period.
- Specify the organizations and individuals who will be responsible for each transition activity, ensuring that responsibility for the software product by the operations and maintenance personnel increases progressively.
- Identify products and support services that will be needed for day-to-day operations or that will enhance operational effectiveness.

Review Process: Conduct a structured walkthrough to assure that the Transition Plan is logical, accurate, and complete. Involve operational and maintenance personnel in the walkthrough.

Activity: **7.8**
Generate Operating Documentation

Responsibility: Project Team/Technical Writer

Description: Plan, organize, and write the operating documentation that describes the functions and features of the software product from the users point-of-view. The different ways that users (including system administration and maintenance personnel) will interact with the software product must be considered. The needs of the users should dictate the document presentation style and level of detail. Responsibilities for changing and maintaining the documents should be described in each document.

The following are typical operating documents for a large software project.

- Users Manual
- Programmers Reference Manual
- Systems Administration Manual
- Data Base Administration Manual
- Operations Manual

It is recommended that a technical writer be involved in the generation of all operating documents. A technical writer works closely with the project team to ensure that documents are grammatically correct; comply with applicable standards; and are consistent, readable, and logical.

Note: The operating documents can be produced as separate manuals or combined to accommodate less complex software projects.

Procedure: Use the following procedure to develop the operating documentation.

- Identify the operating documents that need to be developed. Determine if any of the documents can be combined or delivered as multiple volumes.

- Determine whether the documents should be provided as printed material, standalone electronic files, online documentation accessed through the software product, or a combination.

- Determine the best presentation method or combination of methods required for each of the documents, such as a traditional manual, quick reference guide or card, or online help.

Procedure, continued:

- Identify all of the features of the software user interface and the tasks users will perform.

- Identify the users' needs and experience levels to determine:

  - The amount of user interaction, level of interaction, and whether the interaction is direct or indirect.

  - The appropriate level of detail (e.g., the Users Manual should not contain highly technical terms and explanations that may confuse or frustrate a user).

- Determine the document content and organization based on whether the document will be used more as an instructional tool or a reference guide.

- Develop descriptions of each function and feature of the software product and organize the information to facilitate quick, random access.
- Provide appropriate illustrations and examples to enhance clarity and understanding.

- Establish a schedule for the documents to be reviewed after the software product goes into production. Operating documents must be kept up-to-date as long as the software product remains in production.

Tasks: The following tasks describe the minimum requirements for operating documentation.

7.8.1 Produce Users Manual
7.8.2 Produce Programmers Reference Manual

Task: **7.8.1**
Produce Users Manual

Responsibility: Project Team/Technical Writer

Description: The Users Manual provides detailed information users need to access, navigate through, and operate the software product. Users rely on the Users Manual to learn about the software or to refresh their memory about specific functions. A Users Manual that is organized functionally so that the information is presented the same way the software product works helps users understand the flow of menus and options to reach the desired functions.

Different categories of users may require different types of information. A modular approach to developing the Users Manual to accommodate the needs of different types of users eliminates duplication and minimizes the potential for error or omission during an amendment or update. For example, separate general information that applies to all users from the special information that applies to selected users such as system administrators or data base administrators. The special information can be presented in appendixes or supplements that are only provided to the users who need the information.

Work Product: Write the draft Users Manual in clear, nontechnical terminology that is oriented to the experience levels and needs of the user(s). The following are typical features of a users manual.

- Overview information on the history and background of the project and the architecture, operating environment, and current version or release of the software product.
- Instructions for how to install, setup, or access the software product.
- Complete coverage of all software functions, presented in a logical, hierarchical order.
- Accurate pictures of screens and reports, ideally with data values shown, so the user can easily relate to examples.
- In-depth examples and explanations of the areas of the software product that are most difficult to understand.
- Clear delineation of which features are accessible only to specific users.

Work Product,

continued:

- Instructions on accessing and using online help features.
- Procedures for data entry.
- Descriptions of error conditions, explanations of error messages, and instructions for correcting problems and returning to the function being performed when the error occurred.
- Instructions for performing queries and generating reports.
- Who to contact for help or further information.

Note: For large or complex software products, separate manuals (e.g., User's Manual, Data Base Administrator's Manual, and System Administrator's Manual) may be necessary to address the needs of different categories of users.

For very small projects, a quick reference guide or card may be more appropriate than a full-scale Users Manual. The guide or card should be designed to provide a quick reference of logon, logoff, and commands for frequently used functions.

For projects of any size, a quick reference card may be developed as a supplement to more detailed user documentation.

Review Process: Conduct structured walkthroughs for the draft Users Manual or set of user documents to assure that the documentation is complete, easy to use, and accurately reflects the software product and its functions.

The draft user documentation will be tested and verified with the software product during the Software Integration and Testing Stage.

Task: **7.8.2**
Produce Programmers Reference Manual

Responsibility: Project Team/Technical Writer

Description: The Programmers Reference Manual contains programming information used by the maintenance staff to maintain the programs, data bases, interfaces, and operating environment. The Programmers Reference Manual should provide an overall conceptual understanding of how the software product is constructed and the details necessary to implement corrections, changes, or enhancements.

The Programmers Reference Manual describes the logic used in developing the software product and the functional and system flow to help the maintenance programmers understand how the programs fit together. The information should enable a programmer to determine which programs may need to be modified to change a system function or to fix an error.

Work Product: The following are typical features of a Programmers Reference Manual.

- A description of the technical environment, including versions of the programming language(s) and other proprietary software packages.
- A brief description of the design features including descriptions of unusual conditions and constraints.
- An overview of the software architecture, program structure, and program calling hierarchy.
- The design and programming standards and techniques used to develop the software product.
- Concise descriptions of the purpose and approach used for each program.
- Layouts for all data structures and files used in the software product.
- Descriptions of maintenance procedures, including configuration management, program checkout, and system build routines.
- The instructions necessary to compile, link, edit, and execute all programs.

Work Product, continued:

- Manual and automated backup procedures.
- Error processing features.

Use appendixes to provide detailed information that is likely to change as the software product is maintained. For example, a list of program names and a synopsis of each program could be included as an appendix.

Review Process: Conduct structured walkthroughs of the draft Programmers Reference Manual to assure that the documentation is complete, easy to use, and accurately reflects the software product and its functions.

The draft Programmers Reference Manual will be tested and verified with the software product during the Software Integration and Testing Stage.

Activity: **7.9**
Develop Training Program

Responsibility: Project Team

Description: A Training Program defines the training needed to implement and operate the software product successfully. The Training Plan should address the training that will be provided to the system owner, users, and maintenance staff. When new hardware or software is being used, affected personnel will need hands-on experience before bringing the new equipment or software into daily operation.

Training must address both the knowledge and the skills required to operate and use the system effectively. Design the training program to accomplish the following objectives.

- Provide trainees with the specific knowledge and skills necessary to perform their work.
- Prepare training materials that will sell the software product as well as instruct the trainees. The training program should leave the trainees with the enthusiasm and desire to use the new product.
- Account for the knowledge and skills the trainees bring with them, and use this information as a transition to learning new material.
- Anticipate the needs for follow-on training after the software product is fully operational, including refresher courses, advanced training, and repeats of basic courses for new personnel.
- Build in the capability to update the training as the software product evolves.

Involve the system owner and key users in the planning to determine the education and training needs for all categories of users (managers, users, and maintenance staff).

Work Product: Prepare a draft Training Plan that describes the Training Program and addresses the following issues.

- Identifies personnel to be trained. Review the list of trainees with the system owner and users to ensure that all personnel who should receive training have been identified.

Work Product,

continued:

- Defines the overall approach to training and the required training courses.
- Establishes the scope of the training needed for users, management, operations, and maintenance personnel.
- Define how and when training will be conducted. Specify instructor qualifications, learning objectives, and mastery or certification requirements (if applicable).
- Identify any skill areas for which certification is necessary or desirable. Tailor the training to the certification requirements.
- Establish a preliminary schedule for the training courses. The schedule must reflect training requirements and constraints outside the project. Schedule individual courses to accommodate personnel who may require training in more than one area. Identify critical paths in the training schedule such as the time period for the software product's installation and conversion to production status.
- Define the required course(s), outline their content and sequence, and establish training milestones to meet transition schedules.
- Tailor the instruction methods to the type of material being presented. Include classroom presentation, interactive computer-assisted instruction, demonstrations, individual video presentations, and hands-on experience, either live or simulated.
- Identify trainers who are technically knowledgeable and were involved in the design and development of the system. For projects with extensive and formal training requirements, it may be necessary to provide training for the trainers.
- Consider availability of the following: users, system-tested software, training rooms and equipment, and the completion of system documentation and training materials.

Place a copy of the draft Training Plan in the Project File. The plan will be reviewed and updated during the Software Integration and Testing Stage.

Review Process: Conduct a structured walkthrough to assure that the draft Training Plan is accurate and complete.

Activity: **7.10**
Revise Project Plan

Responsibility: Project Manager

Description: Once the coding effort is completed and unit and integration testing have been conducted, determine if the project estimates for resources, cost, and schedule need to be revised.

Work Product: Review the Project Plan for accuracy and completeness of all Programming Stage activities and make any changes needed to update the information. Expand the information for the Software Integration and Testing Stage to reflect accurate estimates of resources, costs, and hours. Place a copy of the revised Project Plan in the Project File.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Review Process: Conduct a structured walkthrough to ensure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the Software Integration and Testing Stage.

The Project Plan is formally reviewed during the In-Stage Assessment and Stage Exit processes.

Activity: **7.11**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the Programming Stage Exit process. Additional ISAs can be performed during the stage, as appropriate.

Provide the reviewer with copies of all work products developed or revised during the Programming Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/best practices.
- Sound project management practices are being used.
- The project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide*. A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **7.12**
Conduct Programming Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the Programming Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the guide is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the Software Integration and Testing Stage.

Software Engineering Methodology

Chapter 8.0
Software Integration and Testing Stage

Table of Contents

Chapter **Page**

8.0 Software Integration and Testing Stage 8.0-1
- 8.1 Conduct Integration Testing 8.1-1
- 8.2 Conduct System Testing 8.2-1
- 8.3 Initiate Acceptance Process 8.3-1
- 8.4 Conduct Acceptance Test Team Training 8.4-1
- 8.5 Revise Project Plan 8.5-1
- 8.6 Conduct In-Stage Assessment 8.6-1
- 8.7 Conduct Software Integration and Testing Stage Exit 8.7-1

Stage: **8.0**
Software Integration and Testing Stage

Description: Software integration and testing activities focus on interfaces between and among components of the software product, such as functional correctness, system stability, overall system operability, system security, and system performance requirements (e.g., reliability, maintainability, and availability). Software integration and testing performed incrementally provides feedback on quality, errors, and design weaknesses early in the integration process.

In this stage, software components are integrated and tested to determine whether the software product meets predetermined functionality, performance, quality, interface, and security requirements. Once the software product is fully integrated, system testing is conducted to validate that the software product will operate in its intended environment, satisfies all user requirements, and is supported with complete and accurate operating documentation.

Input: The following items provide input to this stage.

- Project File
- Acceptance Test Plan *(draft)*
- Acquisition Plan
- Installation Plan *(draft)*
- Software modules
- Requirements Traceability Matrix *(expanded)*
- Project Test File
- Development baselines
- Transition Plan
- Operating Documentation *(draft)*
  - Users Manual
  - Programmers Reference Manual
- Training Plan *(draft)*
- Integration Test Plan
- System Test Plan
- Project Plan
- Software Quality Assurance Plan

High-Level Activities: The remainder of this chapter is divided into sections that describe specific high-level activities performed during this stage. These activities represent the minimum requirements for a large software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage

High-Level Activities, continued:

requirements to accommodate the different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

8.1 Conduct Integration Testing
8.2 Conduct System Testing
8.3 Initiate Acceptance Process
8.4 Conduct Acceptance Test Team Training
8.5 Revise Project Plan
8.6 Conduct In-Stage Assessment
8.7 Conduct Software Integration and Testing Stage Exit

Output:

Several work products are produced during this stage. The work products listed below are the minimum requirements for a large software project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

- Integration Test Reports
- System Test Report
- Operating Documents *(final)*
  - Users Manual
  - Programmers Reference Manual
- Training Plan *(final)*
- Installation Plan *(final)*
- Acceptance Test Plan *(final)*
- Preacceptance Checklist
- Security Checklist
- Error Reporting and Tracking System *(optional)*
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 8.0-1, Software Integration and Testing Stage Activities and Work Products by Project Size.* The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large projects.

Review Process:

Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. The time and resources needed to conduct the walkthroughs should be indicated in the project resources, schedule, and work breakdown structure.

Reference: *Appendix C, Conducting Structured Walkthroughs,* provides a procedure and sample forms that can be used for structured walkthroughs.

Bibliography: The following materials were used in the preparation of the Software Integration and Testing Stage chapter.

1. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes,* IEEE Std 1074-1991, New York, 1992.

2. U.S. Department of Commerce, National Institute of Standards and Technology, *Guide to Software Acceptance,* 500-180, Washington, D.C., 1990.

3. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual,* 1988.

4. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual,* 1988.

Exhibit 8.0-1. Software Integration and Testing Stage Activities and Work Products by Project Size

Work Activity		Project Size			Work Product	Scheduled Deliverables		
		L	M	S		L	M	S
8.1	Conduct Integration Testing	R	R	R	Integration Test Materials	R	R	R
					Integration Test Report	R	R	R
8.2	Conduct System Testing	R	R	R	System Test Materials	R	R	R
					System Test Report	R	R	R
					Operating Documents *(final)*	R	R	R
					Training Plan *(final)*	R	R	R
					Installation Plan *(final)*	R	R	A
8.3	Initiate Acceptance Process	R	R	R	Acceptance Test Plan *(final)*	R	R	R
					Preacceptance Checklist	R	R	R
					Security Checklist	A	A	A
					Error Reporting and Tracking System *(optional)*	O	O	O
8.4	Conduct Acceptance Test Team Training	A	A	A				
8.5	Revise Project Plan	R	R	A	Project Plan *(revised)*	R	R	A
8.6	Conduct In-Stage Assessment	R	R	A	ISA Report Form[1]	N	N	N
8.7	Conduct Software Integration and Testing Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
S = Small

Minimum Requirements: R = Required
A = As Appropriate
N = Not Applicable

O = Optional work product
[1] = Completed by reviewer

Activity: **8.1**
Conduct Integration Testing

Responsibility: Project Team Programmers

Description: During software integration, the software components developed by the programming staff, off-the-shelf software purchased from vendors, and reusable code or modules obtained from other sources are assembled into one software product. Each assembly is tested in a systematic manner in accordance with the Integration Test Plan. An incremental approach to integration enables verification that as each new component is integrated, it continues to function as designed and both the component and the integrated product satisfy their assigned requirements.

Integration testing is a formal procedure that must be carefully planned and coordinated with the completion dates of the unit-tested modules. Integration testing begins with a software structure where called sub-elements are simulated by stubs. A stub is a simplified program or dummy module designed to provide the response (or one of the responses) that would be provided by the real sub-element. A stub allows testing of calling program control and interface correctness. Stubs are replaced by unit-tested modules or builds as integration testing proceeds. This process continues one element at a time until the entire system has been integrated and tested.

Integration testing may be performed using "bottom up" or "top down" techniques. Most integration test plans make use of both bottom-up and top-down techniques. Scheduling constraints and the need for parallel testing will affect the test approach.

The bottom-up approach incorporates one or more modules into a build; tests the build; and then integrates the build into the software structure. The build normally comprises a set of modules that perform a major function of the software system. Initially, the function may be represented by a stub that is replaced when the build is integrated.

In the top-down approach, individual stubs are replaced so that the top-level control is tested first, followed by stub replacements that move downward in the software structure. Using top-down integration, all modules that comprise a major function are integrated, thereby allowing an operational function to be demonstrated prior to completion of the entire system.

Work Products: At the completion of each level of integration testing, a test report is written. The report documents test results and lists any discrepancies that must be resolved before the tested components can be used as the foundation for another integration level. Place a copy of all integration test materials in the Project Test File.

A final test report is generated at the completion of integration testing indicating any unresolved difficulties that require management attention. Place a copy of the final Integration Test Report in the Project File.

Optional Work Product: A formal reporting system by which detected errors and discrepancies are recorded and fully described is recommended. These reports will help to confirm that all known errors are fixed before delivery of the completed software product. Error reports also help to trace multiple instances of the same error or anomalous behavior, so that error correction and prevention assignments can be implemented. The Quality Assurance representative assigned to the project can provide assistance in developing and using an error reporting/tracking system.

Activity: **8.2**
Conduct System Testing

Responsibility: Project Team or Independent Test Team

Description: During system testing, the completely integrated software product is tested to validate that the product meets all requirements. System response timing, memory, performance, security, and the functional accuracy of logic and numerical calculations are verified under both normal and high-load conditions. Query and report capabilities are exercised and validated. All operating documents are verified for completeness and accuracy.

System testing is conducted on the system testbed using the methodology and test cases described in the System Test Plan. The system testbed should be as close as possible to the actual production system. Either the project team or an independent test team conducts system testing to assure that the system performs as expected and that each function executes without error. The results of each test are recorded and upon completion included as part of the project test documentation.

When errors are discovered, they should be reviewed by the test team leader to determine the severity and necessary subsequent action. If appropriate, minor problems can be corrected and regression tested by the project team programmers within the time frame allotted for the system test. Any corrections or changes to the software product must be controlled under configuration management. Major problems may be cause to suspend or terminate the system test, which should then be rescheduled to begin after all of the problems are resolved.

Encourage users to participate in the system tests to gain their confidence in the software product and to receive an early indication of any problems from the user's perspective. Inform users that errors and discrepancies may occur during testing and explain the error correction, configuration management, and retest processes.

At the successful conclusion of system testing, the software product is ready for installation and acceptance testing.

Work Products: Review the draft versions of the operating documents, Training Plan, and Installation Plan. Update the documents as needed. Deliver the final versions of the operating documents, Training Plan, and Installation Plan to the system

Work Products, continued:

owner and user for review and approval. Place a copy of the approved documents in the Project File. Place a copy of all system test materials (e.g., inputs, outputs, results, and error logs) in the Project Test File.

Generate a test report at the conclusion of the system test process. The report documents the system test results and lists any discrepancies that must be resolved before the software product is installed and prepared for acceptance testing. Place a copy of the report in the Project File.

Activity: **8.3**
Initiate Acceptance Process

Responsibility: Project Manager

Description: The acceptance process is used to officially accept new or modified software products that satisfy the project requirements and are fully operational. The initiation of the acceptance process begins after the successful completion of system testing. Prior to the initiation of the acceptance process, review the draft Acceptance Test Plan. Make any additions or changes needed to assure that the test plan reflects the current version of the software requirements.

The acceptance process is initiated with the completion of a Preacceptance Checklist. This list helps to ensure that all necessary preacceptance activities have been completed and that the required operating documents were developed and approved. The Preacceptance Checklist includes a section on software security issues.

Procedure: Use the following procedure to initiate the acceptance process.

- The project manager notifies the Quality Assurance Team assigned to the project that the project is ready to start the acceptance process.
- The Quality Assurance Team sends the Preacceptance Checklist to the project manager.
- The project manager completes the checklist, obtains the concurrence signature of the system owner (if required), and returns the completed checklist to the Quality Assurance Team.
- The Quality Assurance Team schedules an initial acceptance process meeting. More than one meeting may be necessary to accommodate users at different locations or with varying requirements.

Work Product: Review the draft version of the Acceptance Test Plan, and update as needed. Deliver the final version of the Acceptance Test Plan to the system owner, user, and other project stakeholders for review and approval prior to conducting any acceptance tests. Place a copy of the approved Acceptance Test Plan in the Project File. The Preacceptance Checklist is completed and submitted to the Quality Assurance Team supporting the project. A sample Preacceptance Checklist is provided on the following pages.

Sample Preacceptance Checklist
[Software Product Name]

Instructions: The project manager must indicate completion of each item with a checkmark and obtain concurrence on the last page. Send a copy of the completed checklist to the Quality Assurance Team assigned to the project. Any deviations from the checklist must be documented and a copy attached to the checklist.

1. Acceptance Test Plan

__ Provided to Quality Assurance and approved.

2. System Documents (e.g., User's Manual and Programmer's Reference Manual)

__ Appropriate system documents have been prepared in accordance with the applicable documentation standards. The system documents have been reviewed and approved by the system owner and other designated approvers. Issues identified during the Stage Exits have been resolved.

3. System Review Inventory System (SRIS)

__ The SRIS form has been updated to reflect the latest version of the software product and was submitted to the site software inventory/repository administrator and the Headquarters SRIS Coordinator.

4. Quality Assurance Support Preparation

__ The Quality Assurance Team has been given access to all portions of the software product required for testing.

__ Software product information has been provided to the Quality Assurance Team.

5. Security

__ The security checklist has been completed by the system owner and forwarded to the site's Computer Protection Program Manager. A sample security checklist is provided at the end of the Preacceptance Checklist.

__ Criteria for determination of mission essentiality have been reviewed and a determination made for the software product.

The software is mission essential. ☐ Yes ☐ No

__ If mission essential, a Continuity of Operations Plan is in place. The plan has been reviewed and approved by the appropriate organizations and personnel.

6. General Requirements

__ Installation of the required hardware and software for all users is fully documented in an Installation Plan.

__ User training is fully documented in an approved Training Plan.

__ Maintenance programmer staff training is fully documented in an approved Training Plan.

__ Configuration management is occurring according to the Software Configuration Management Plan.

__ Data retention criteria have been established according to appropriate recordkeeping requirements.

__ User identification has been assigned.

__ A task assignment has been generated to cover enhancement and maintenance services after acceptance.

Sample Preacceptance Checklist (continued)
[Software Product Name]

7. Software Requirements

__ All debugging and monitoring facilities have been removed from the production source and load (executable) modules.

__ Appropriate operational area points-of-contact have been consulted and agree that all operational readiness issues have been satisfied.

8. System Testing

System testing has been performed on all programs and modules to verify that the following conditions have been met.

__ User-required features have been satisfied (e.g., reports, data entry, data validation, queries).

__ All error conditions specified in the Integration and System Test Plans have been tested and respond to corrective action.

__ All backup, recovery, checkpoint, purge, and restart facilities required to ensure system integrity are operational.

__ Response times have been demonstrated and are in line with the requirements.

__ A production data base has been established.

System Owner Concurrence (if required)

I concur that all of the above items have been completed, and the system is ready for the acceptance process. Any deviations from the checklist have been documented and approved.

_____
System Owner

Sample Preacceptance Checklist
Security Issues
[Software Product Name]

1. Sensitivity and Essentiality

The software is: ☐ non-sensitive ☐ sensitive
☐ classified ☐ mission essential

2. General Security (all software products)

	Yes	No	
a.	☐	☐	Security objectives were established by the system owner.
b.	☐	☐	Security requirements were specified by the project team to support the security objectives.
c.	☐	☐	System design features enforce the security requirements.
d.	☐	☐	Testing was conducted to verify the security design features incorporated into the software product and the results were recorded.
e.	☐	☐	Security tests were conducted satisfactorily or a statement of acceptance of risk was issued by the system owner.
f.	☐	☐	Appropriate data set/file protection rules, authorities, and user identification codes were established by the system owner or as mandated by higher authority.
g.	☐	☐	Access control protection was incorporated into the software product.
h.	☐	☐	All manufacturer generic, test team, temporary, and superfluous passwords were deleted from the software product.
i.	☐	☐	All privacy, freedom of information, sensitivity, and classification considerations were identified, resolved, and established.

3. Classified Software Products

	Yes	No	
a.	☐	☐	An approved Security Plan was developed.
b.	☐	☐	For applications running on a classified system, the system owner has provided the required information to the Computer System Security Officer (CSSO) for the Computer Center for inclusion in the security plan.
c.	☐	☐	The security test plan was approved.
d.	☐	☐	The security test was successfully completed.
e.	☐	☐	The software product was certified by the owning organization's CSSO.

Activity: **8.4**
Conduct Acceptance Test Team Training

Responsibility: Project Team

Description: If the project team is not conducting the Acceptance Test, training may be required for the personnel performing the testing. The acceptance test participants and their experience with the software product and the operating environment should have been identified in the Acceptance Test Plan.

The level of training will depend on the testers' familiarity with the software product and the platform on which the software will run. The advantage of having users acceptance test the software product is that they are the experts most familiar with the business information flow and how the software product must fit into the workplace.

It is recommended that the operating documents and other test materials be distributed to the test team prior to the actual start of the acceptance test training. This will give the test team time to become familiar with the software product and the test process and procedures.

Activity: **8.5**
Revise Project Plan

Responsibility: Project Manager

Description: Once the integration and system tests are completed, determine if the project estimates for resources, cost, and schedule need to be revised.

Work Product: Review the Project Plan for accuracy and completeness of all Software Integration and Testing Stage activities and make any changes needed to update the information. Expand the information for the Installation and Acceptance Stage to reflect accurate estimates of resources, costs, and hours. Place a copy of the revised Project Plan in the Project File.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Review Process: Conduct a structured walkthrough to ensure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the Installation and Acceptance Stage.

The Project Plan is formally reviewed during the In-Stage Assessment and Stage Exit processes.

Activity: **8.6**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the Software Integration and Testing Stage Exit process. Additional ISAs can be performed during the stage, as needed. Periodic reviews of the integration and system test results and logs are recommended.

Provide the reviewer with copies of all work products developed or revised during the Software Integration and Testing Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/best practices.
- Sound project management practices are being used.
- Project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide.* A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **8.7**
Conduct Software Integration and Testing Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the Software Integration and Testing Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the guide is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the Installation and Acceptance.

Software Engineering Methodology

Chapter 9.0
Installation and Acceptance Stage

Table of Contents

Chapter **Page**

9.0 Installation and Acceptance Stage 9.0-1
- 9.1 Perform Installation Activities 9.1-1
- 9.2 Conduct Installation Tests 9.2-1
- 9.3 Conduct User Training 9.3-1
- 9.4 Conduct Acceptance Test 9.4-1
- 9.5 Conclude Acceptance Process 9.5-1
- 9.6 Transition to Operational Status 9.6-1
- 9.7 Revise Project Plan 9.7-1
- 9.8 Conduct In-Stage Assessment 9.8-1
- 9.9 Conduct Installation and Acceptance Stage Exit 9.9-1

Chapter: **9.0**
Installation and Acceptance Stage

Description: Installation and acceptance of the software product are initiated after the system test has been successfully completed. This stage involves the activities required to install the software, data bases, or data that comprise the software product onto the hardware platform at the site(s) of operation. The objectives of the activities in this stage are to verify that the software product meets design requirements and to obtain the system owner's acceptance and approval of the software product. The activities associated with this stage should be performed each time the software product is installed at an acceptance test site or production site.

User training may be required to complete the installation process. A description of the training necessary for programmers, testers, users, and operations staff is provided in the Training Plan.

Input: The following items provide input to this stage:

- Integration Test Materials
- System Test Materials
- Operating Documents
  - Users Manual
  - Programmers Reference Manual
- Training Plan
- Installation Plan
- Conversion Plan
- Acceptance Test Plan
- Preacceptance Checklist
- Security Checklist
- Project Plan *(revised)*
- Software Quality Assurance Plan

High-Level Activities: The remainder of this chapter is divided into sections that describe the specific high-level activities performed during this stage. These activities represent the minimum requirements for a large software engineering effort. *Notes* are provided, as applicable, to assist in customizing these lifecycle stage requirements to accommodate the different sizes of software engineering efforts. The high-level activities are presented in the sections listed below.

9.1 Perform Installation Activities
9.2 Conduct Installation Tests

High-Level Activities, continued:

9.3 Conduct User Training
9.4 Conduct Acceptance Test
9.5 Conclude Acceptance Process
9.6 Transition to Operational Status
9.7 Revise Project Plan
9.8 Conduct In-Stage Assessment
9.9 Conduct Installation and Acceptance Stage Exit

Output: Several work products are produced during this stage. The work products listed below are the minimum requirements for a large software project. Deviations in the content and delivery of these work products are determined by the size and complexity of the project. Explanations of the work products are provided under the applicable activities described in the remainder of this chapter.

- Converted data or system files
- Installation Test materials
- User Training materials
- Acceptance Test Report
- Acceptance Checklist
- Operational software product
- Operating documents
- Project Plan *(revised)*

A matrix showing the work products associated with each high-level activity is provided in *Exhibit 9.0-1, Installation and Acceptance Stage Activities and Work Products*. The matrix also shows which work products are deliverables and whether they are required or optional for small, medium, and large projects.

Review Process: Structured walkthroughs are necessary during this stage to validate work products. The activities that are appropriate for structured walkthroughs are identified throughout the chapter. The time and resources needed to conduct the walkthroughs should be indicated in the project resources, schedule, and work breakdown structure.

Reference: *Appendix C, Conducting Structured Walkthroughs*, provides a procedure and sample forms that can be used for structured walkthroughs.

Exhibit 9.0-1. Installation and Acceptance Stage Activities and Work Products by Project Size

Work Activity		Size L	Size M	Size S	Work Product	Scheduled Deliverables L	Scheduled Deliverables M	Scheduled Deliverables S
9.1	Perform Installation Activities	R	R	R	Converted data or system files	I	I	I
9.2	Conduct Installation Tests	R	R	R	Installation Test materials	R	R	R
9.3	Conduct User Training	R	R	R	User training materials	R	R	R
9.4	Conduct Acceptance Test	R	R	R	Acceptance Test materials Acceptance Test Report	R R	R R	R R
9.5	Conclude Acceptance Process	R	R	A	Acceptance Checklist	R	R	A
9.6	Transition to Operational Status	R	R	R	Operational software product Operating documents	R R	R R	R R
9.7	Revise Project Plan	R	R	R	Project Plan (revised)	R	R	A
9.8	Conduct In-Stage Assessment	R	R	A	ISA Report[1]	N	N	N
9.9	Conduct Installation and Acceptance Stage Exit	R	R	A	Stage Exit Meeting Summary	N	N	N

Size: L = Large
M = Medium
S = Small

Minimum Requirements: R = Required
A = As Appropriate
N = Not Appropriate

I = Input to other deliverables
[1] = Completed by reviewer

Bibliography: The following materials were used in the preparation of the Installation and Acceptance Stage chapter.

1. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Developing Software Life Cycle Processes*, IEEE Std 1074-1991, New York, 1992.

2. U.S. Department of Commerce, National Institute of Standards and Technology, *Guide to Software Acceptance*, 500-180, Washington, D.C., 1990.

3. U.S. Department of Energy, *Automated Data Processing Systems Development Methodology, Volume 1*, K/CSD/INF/86-3, Vol.1,R3, prepared under contract by Martin Marietta Energy Systems, Inc, at Oak Ridge National Laboratory, August 1987.

4. U.S. Department of Energy, *DOE/NV Software Management Plan*, Nevada Operations Office, May 1991.

5. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual*, 1988.

6. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Standards Manual*, 1988.

Activity: **9.1**
Perform Installation Activities

Responsibility: Project Team

Description: The installation process involves loading, copying, or migrating the software and data, if required, to the production platform and the provision of operating documentation and other support materials at each site. The installation of firmware, hardware, and telecommunications equipment may also be involved.

If a current system exists, implement system and data conversion in accordance with the procedures described in the Conversion Plan. Each data and file conversion should include a confirmation of data and file integrity. Determine what the output in the new software product should be compared with the current system, and assure that the data and files are synchronized.

At each installation site, inspect the facility to assure that site preparation is complete and in accordance with the Installation Plan. Initiate any actions that are needed to complete the preparations. Conduct an inventory of all vendor provided hardware, software, firmware, and telecommunications equipment in accordance with the Acquisition Plan.

Follow the procedure specified in the Installation Plan when installing the software, hardware, and other equipment. Monitor all installation activities including those performed by vendors.

Procedure: Use the following procedure to perform the installation activities.

- Coordinate the installation with the system owner, users, operations staff, and other affected organizations.
- Ensure that any necessary modifications to the physical installation environment are completed.
- Inventory and test the hardware that will support the software product. This inventory should be performed in advance of the planned installation date to allow time for missing hardware to be obtained and malfunctioning equipment to be replaced or repaired.
- If the software product requires an initial data load or data conversion, install and execute the tested programs to perform this process.
- Install the software product onto the hardware platform.

Activity: **9.2**
Conduct Installation Tests

Responsibility: Project Team

Description: Ensure the integrity and quality of the installed software product by executing the installation tests defined in the Installation Plan. Testing is performed to verify that the software product has been properly installed and is fully operational.

The installation test(s) are designed to validate all functions of the software product and should specify a standard set of test results and tolerances. If the software product being installed is a modification to an existing system, all remaining functions and code that may be affected by the new software should be tested.

Document any problems and identify corrective action. Select a diagnostic package that will pinpoint problems quickly and allow for timely corrections. Retest all equipment and software after a repair, replacement, or modification. Certify each software component on successful completion of installation and checkout. When installation is complete, rerun a portion or all of the system test and dry-run the acceptance test procedures to verify correct operation of the software product.

Conduct installation testing to verify the following:

- Security functions
- Integration with the current software
- Interfaces with other systems
- System functionality based on the requirements

Work Product: Place a copy of all Installation Test materials in the Project File.

Activity: **9.3**
Conduct User Training

Responsibility: Project Team

Description: User training is an important factor in the success of the operational software product. During training, most users will receive their first hands-on experience with the software product. Operations and maintenance staff may also be trained to use, monitor, and maintain the software product. The objective of the training is to provide the trainee with the basic skills needed to effectively use the software product and to raise the user's confidence and satisfaction with the product.

The type of training will depend on the complexity of the software product, and the number and location of the users to be trained. Alternative training formats include formal classroom training, one-on-one training, computer-based instruction, and sophisticated help screens and online documentation. Conduct the training as described in the Training Plan.

Consider conducting a pilot test of the training session(s). Include members of the project team, the system owner, and key users. Have all participants evaluate the training content, instruction, and support materials. Make any necessary changes to the training program prior to general user training sessions.

If consecutive training classes are conducted, feedback should be requested from the participants and used to continuously improve the training approach, methods, and materials.

At the completion of the training, users should have a thorough understanding of the input requirements of each transaction, the processing that takes place, and the types of output that are generated.

Work Product: Submit a copy of the training materials to the system owner and user for review and approval. Place a copy of the approved training materials in the Project File. Training materials are subject to the same configuration control procedures as the other operating documents and should remain current with changes to the software product.

Activity: **9.4**
Conduct Acceptance Test

Responsibility: Acceptance Test Team

Description: Acceptance of a delivered software product is the ultimate objective of a software development project. Acceptance testing is used to demonstrate the software product's compliance with the system owner's requirements and acceptance criteria.

At the system owner's discretion, acceptance testing may be performed by the project team, by the system owner and users with support from the project team, or by an independent verification and validation team. Whenever possible, users should participate in acceptance testing to assure that the software product meets the users' needs and expectations. All acceptance test activities should be coordinated with the system owner, user(s), operations staff, and other affected organizations.

Acceptance testing is conducted in the production environment using acceptance test data and test procedures established in the Acceptance Test Plan. Testing is designed to determine whether the software product meets functional, performance, and operational requirements. If acceptance testing is conducted on an incremental release basis, the testing for each release should focus on the capabilities of the new release while verifying the correct operation of the requirements incorporated in the previous release.

Acceptance testing usually covers the same requirements as the system test. Acceptance testing may cover additional requirements that are unique to the operational environment. The results of each test should be recorded and included as part of the project test documentation.

Subject the test environment to strict, formal configuration control to maintain the stability of the test environment and to assure the validity of all tests. Review the acceptance test environment, including the test procedures and their sequence, with the system owner and user before starting any tests.

Testing is complete when all tests have been executed correctly. If one or more tests fail, problems are documented, corrected, and retested. If the failure is significant, the acceptance test process may be halted until the problem is corrected.

Work Product: Prepare a formal Acceptance Test Report. Summarize the test procedures executed, any problems detected and corrected, and the projected schedule for correcting any open problem reports. Place a copy of all acceptance test materials in the Project Test File.

Review Process: Conduct an Operational Readiness Review at the completion of acceptance testing. This review is a combined quality assurance and configuration management activity. It focuses on the results of the acceptance test and the readiness of the software product to go into production.

During the Operational Readiness Review examine acceptance test results with the system owner and user. Document any problems, determine solutions to the problems, and implement action plans. Once any problems associated with the acceptance test are resolved, the software product is ready for formal acceptance by the system owner.

A successful Operational Readiness Review establishes the operational baseline for the software product. The baseline consists of the software product and the technical documentation that describes the operational software and its characteristics.

Activity: **9.5**
Conclude Acceptance Process

Responsibility: Project Manager

Description: The acceptance process is used to officially accept new or modified software products that satisfy the users' requirements and are fully operational. The acceptance process is concluded when the acceptance test has been successfully completed, the software product has been installed and is operational at all user sites, and complete operating documentation describing the product has been approved and delivered.

At the conclusion of the acceptance process, responsibility for the software product is formally transferred from the project team to the system owner and maintenance staff.

Procedure: Use the following procedure to conclude the acceptance process.

- The project manager notifies the Quality Assurance representative assigned to the project that the software product is ready to complete the acceptance process.
- The Quality Assurance representative sends the Acceptance Checklist to the project manager.
- The project manager completes the checklist, obtains the concurrence signature of the system owner (if required), and returns the completed checklist to the Quality Assurance representative.
- The Quality Assurance representative schedules an acceptance meeting. More than one meeting may be necessary to accommodate users at different locations or with varying requirements.

Work Product: The Acceptance Checklist is completed and submitted to the Quality Assurance representative supporting the project. This list helps to ensure that all necessary acceptance activities have been completed and that the required operating documents were developed and approved. A sample Acceptance Checklist is provided on the following page.

A formal written acceptance of the software product is produced by the system owner to verify that the software product is acceptable and ready for production.

Sample Acceptance Checklist
[Software Product Name]

The Acceptance Checklist provides a final check to determine that everything is in place prior to the system being turned over to the system owner and maintenance staff.

Instructions: The project manager must indicate completion of each item with a checkmark, obtain the concurrence signature of the system owner, and return the completed checklist to the Quality Assurance representative assigned to the project. Any deviations from the checklist must be documented and a copy attached to the checklist.

Yes	**Not Applicable**	
☐	☐	Maintenance programmer data base access has been assigned.
☐	☐	File protection rules have been modified to permit access to the system by maintenance programmers.
☐	☐	File protection rules have been modified to remove project team and other temporary user access from further access to the system.
☐	☐	Any software installed on testbeds or other platforms for acceptance testing has been removed and any file modifications such as to autoexec.bat and config.sys have been restored.
☐	☐	Project team and other individuals having temporary data base access passwords have been deleted.
☐	☐	Final operating documentation has been distributed.
☐	☐	List of any planned enhancements has been sent to the Quality Assurance representative.
☐	☐	Programs, files, and other support software are in the production library and have been deleted from the test library, where appropriate.
☐	☐	All current program compiles, files, and other software have been reviewed and turned over to the maintenance staff.
☐	☐	Load library matches source library.

System Owner Concurrence (if required)

I concur that all of the above items have been completed, the system is acceptable as delivered, and the system is ready to be turned over to the maintenance staff. Any deviations from this checklist have been documented and approved.

_____
System Owner

Activity: **9.6**
Transition to Operational Status

Responsibility: Transition Team

Description: The transition of the software product to full operational status begins after the formal acceptance by the system owner. Use the procedures described in the Transition Plan to implement the transition processes. Conduct or support stress tests and other operational tests. Determine product tolerances to adverse conditions, failure modes, recovery methods, and specification margins. Complete any training and certification activities. Ensure that support to be provided by contractors begins as planned.

The project team is usually expected to provide operational and technical support during the transition. Identify project team personnel with a comprehensive understanding of the software product who can provide assistance in the areas of software installation and maintenance, test, and documentation of changes. Technical support may involve the analysis of problems in software components and operational procedures, the analysis of potential enhancements, and vendor-supplied upgrades to software components (such as the operating system or data base management system).

Transition to full operational status should be an event-oriented process that is not complete until all transition activities have been successfully performed. Withdraw the support of the project team personnel in a gradual sequence to ensure the smooth operation of, and user confidence in, the software product. At the conclusion of the transition process, plan a formal transfer of all responsibility to the maintenance staff. All Project File materials, operating documents, and other pertinent records should be turned over to the maintenance staff at this time.

For major software systems involving multiple organizations and interfaces with other systems, a formal announcement of the transition to production is recommended. The announcement should be distributed to all affected groups. The names and telephone numbers of the maintenance staff should be included.

Activity: **9.7**
Revise Project Plan

Responsibility: Project Manager

Description: Once system installation and acceptance are complete, determine if the project estimates for resources, cost, and schedule need to be revised.

Work Product: Review the Project Plan for accuracy and completeness of all Installation and Acceptance Stage activities and make any changes needed to update the information. Expand the information for the Maintenance Stage to reflect accurate estimates of resources, costs, and hours. Place a copy of the revised Project Plan in the Project File.

Note: A Project Plan is an effective management tool that is recommended for all projects regardless of size. The plan can be consolidated for small projects.

Review Process: Conduct a structured walkthrough to ensure that the Project Plan reflects the project's current status and adequately estimates the resources, costs, and schedule for the Maintenance Stage.

The Project Plan is formally reviewed during the In-Stage Assessment and Stage Exit processes.

Activity: **9.8**
Conduct In-Stage Assessment

Responsibility: Project Manager and Independent Reviewer

Description: An In-Stage Assessment (ISA) is an independent review of the work products and deliverables developed or revised during each stage of the project lifecycle. The independent reviewer is typically a member of the Quality Assurance Team who is assigned to the software project and conducts all of the ISAs for the project.

An ISA does not require meetings with, or extra work by, the project team. All of the work products and deliverables needed for the review should be readily available in the Project File.

Schedule at least one ISA prior to the Installation and Acceptance Stage Exit process. Additional ISAs can be performed during the stage, as needed.

Provide the reviewer with copies of all work products developed or revised during the Installation and Acceptance Stage including the Project Plan. The reviewer assesses the work products and deliverables to verify the following:

- The project is complying with the site's software engineering standards/ best practices.
- Sound project management practices are being used.
- Project risks are identified and mitigated.

A description of the ISA process and the ISA report form are provided in the *In-Stage Assessment Process Guide.* A copy of the guide is provided in Appendix D.

Note: An ISA is an effective project management tool that is recommended for all projects regardless of size.

Work Product: An ISA report form is prepared by the independent reviewer and is used to identify open issues that need to be resolved in this stage. The report is delivered to the project manager and a copy should be placed in the Project File.

Activity: **9.9**
Conduct Installation and Acceptance Stage Exit

Responsibility: Project Manager

Description: The Stage Exit is a process for ensuring that projects are on target, within budget, on schedule, and meet the DOE and project standards identified in the Project Plan. The goal of a Stage Exit is to secure the approval of designated key individuals to continue with the project and to move forward into the next lifecycle stage.

Schedule the Stage Exit as the last activity of the Installation and Acceptance Stage. It is the responsibility of the project manager to notify the appropriate participants when a project is ready for the Stage Exit process and to schedule the Stage Exit meeting. All functional areas and the Quality Assurance representative involved with the project should receive copies of the work products and deliverables produced in this stage.

During the Stage Exit meeting, participants discuss open issues that will impact the Project Plan. The project manager should ensure that an acceptable action plan is developed for handling all open issues. At the conclusion of the meeting, concurrence is needed from the designated approvers to begin the next stage.

A description of the Stage Exit process is provided in the *Stage Exit Process Guide*. A copy of the guide is provided in Appendix E.

Note: A Stage Exit is an effective project management tool that is recommended for all software projects regardless of size. For small software projects, stages can be combined and addressed during one Stage Exit.

Work Product: A summary of the Stage Exit meeting is prepared by the project manager or a designee and distributed to the meeting attendees. The summary identifies any issues and action items needed to obtain concurrence prior to proceeding to the Maintenance Stage.

Software Engineering Methodology

Chapter 10.0
Software Maintenance

Table of Contents

Chapter **Page**

10.0 Software Maintenance 10.0-1
- 10.1 Problem/Modification Identification Stage 10.1-1
- 10.2 Analysis Stage 10.2-1
- 10.3 Design Stage 10.3-1
- 10.4 Programming Stage 10.4-1
- 10.5 System Test Stage 10.5-1
- 10.6 Acceptance Stage 10.6-1
- 10.7 Delivery Stage 10.7-1

Chapter: **10.0**
Software Maintenance

Description: This chapter describes an iterative process for conducting software maintenance activities. The process prescribes a minimal set of criteria that are necessary for project management and quality assurance processes; control; and management of the planning, execution, and documentation of software maintenance activities. The use of automated tools to facilitate requirements definition, design, coding, and system documentation is encouraged. The selection and implementation of tools varies among sites and organizations.

The basic maintenance process model includes input, process, output, and control for software maintenance. It is based on the same software engineering principles and preferred practices that lower risk and improve quality during the planning and development stages of the lifecycle. The process model supports the concept that planned software changes should be grouped and packaged into scheduled releases that can be managed as projects. This proven approach allows the maintenance team to better plan, optimize use of resources, take advantage of economies of scale, and better control outcome in terms of both schedule and product quality.

Each organization performing software maintenance activities should have a local documented procedure for handling emergency changes that cannot be implemented as part of a scheduled release. Generally, these changes include fixes to correct defects and updates to meet unscheduled business or legal requirements. Emergency changes should be integrated into the next release for full regression testing and documentation updates.

Stages: The activities to be performed during software maintenance are grouped into logically related segments of work called "stages." These stages are similar to those referenced in the planning and development stages of the software lifecycle. The stages are presented in the sections listed below.

10.1 Problem/Modification Identification Stage
10.2 Analysis Stage
10.3 Design Stage
10.4 Programming Stage
10.5 System Test Stage
10.6 Acceptance Stage
10.7 Delivery Stage

A matrix depicting the maintenance process model stages is provided in *Exhibit 10.0-1, Process Model for Software Maintenance.*

Note: The maintenance process model does not presume the use of any particular software development methodology (e.g., waterfall or spiral). This process is valid regardless of size, complexity, criticality, application of the software product, or usage of the software in the system to be maintained.

The software maintenance stages can be tailored (i.e., logically combining stages or outputs) as appropriate. Stages may be combined to more effectively manage the project. Decisions to combine stages are agreed to by the designated approvers during the Analysis Stage. Factors that can influence the number of stages include level of effort, complexity, visibility, and business impact. Guidance to assist with decisions to combine stages is presented in *Exhibit 10.0-2, Tailoring For Size*.

Project Management: To the extent possible, all software maintenance activity should be managed as a project to gain the benefits inherent in project management and to enable tracking of activities and costs. The extent of project management activity will vary, and should be tailored according to the size, complexity, and impact of the change or enhancement.

Review Processes: In each stage, one or more structured walkthroughs are conducted to validate work products. *Appendix C, Conducting Structured Walkthroughs*, provides a procedure and sample forms that can be used for structured walkthroughs.

In software maintenance, and especially for major modifications, one or more In-Stage Assessments are conducted as part of the quality assurance activities for each stage. This process is documented in *Appendix D, In-Stage Assessment (ISA) Process Guide*.

A Stage Exit is conducted at the end of each stage of software maintenance. This process, which includes definition of participant roles and the review and approval process, is documented in *Appendix E, Stage Exit Process Guide*.

Metrics: Metrics/measures and associated factors for each stage should be collected and reviewed at appropriate intervals. *Exhibit 10.0-3, Process Model Metrics for Software Maintenance,* provides metrics for each stage of software maintenance. Metrics/measures captured for maintenance should enhance the implementation and management of this process.

Conventions: The following convention is used in each exhibit depicting a software maintenance stage.

```
                    Control
                       T
                       *
                       ▾
                 +)))))))))))),
Input S))))▸ *Process Name* S)))▸ Output
                 .)))))))))))-
                       T
                       *

              Associated Process
```

An "associated process" is one that is executed in support of software maintenance, but is itself not defined in this chapter (e.g., Stage Exit).

Exhibit 10.0-1. Process Model for Software Maintenance

	Problem Identification Stage	Analysis Stage	Design Stage	Programming Stage	System Test Stage	Acceptance Stage	Delivery Stage
Input	Modification Request (MR)	Project/system document Project file information Validated MR	Project/system document Source code Database(s) Analysis stage output	Source code Product/system document Results of design stage	Updated software Documentation Test Readiness Review Report Updated system	Test Readiness Review Report Integrated system Acceptance test: Plans Cases Procedures	Tested/accepted system
Process	Assign change number Classify Accept or reject change Preliminary effort estimate	Feasibility analysis Detailed analysis Redocument, if needed	Revise: Requirements System doc. Module doc. Project plan Create test cases	Code Unit test Test Readiness Review	Functional test Interface testing Regression testing Test Readiness Review	Acceptance test Interoperability test Functional Configuration Audit (FCA)	Physical Configuration Audit (PCA) Install Training
Output	Validated MR Process determinations	Feasibility Report Detailed Analysis Report Updated: Requirements Modification list Test strategy Project plan	Revised: Modification list Detailed analysis Updated: Design baseline Test plans Project plan	Updated: Software Design documents Test documents User documents Training materials Project plan Test readiness review report	Tested system Test reports Updated project plan	New system baseline Acceptance Test Report FCA Report Updated project plan	PCA Report Version Description Document (VDD)
Review Assurance Approve	Peer review(s)	Structured Walkthrough(s) In-Stage Assessment(s) Stage Exit	Structured Walkthrough(s) In-Stage Assessment(s) Stage Exit	Structured Walkthrough(s) In-Stage Assessment(s) Stage Exit	Structured Walkthrough(s) In-Stage Assessment(s) Stage Exit	Structured Walkthrough(s) In-Stage Assessment(s) Stage Exit	Structured Walkthrough Stage Exit
Metrics	*See Exhibit 10.0-3, Process Model Metrics for Software Maintenance*						

Exhibit 10.0-2. Tailoring For Size

```
                                SOFTWARE MAINTENANCE STAGES
 SIZE[1]
                        ▾         ▾         ▾         ▾         ▾         ▾         ▾

            *         *         *         *         *         *         *         *
LARGE       /)))))))))3)))))))))3)))))))))3)))))))))3)))))))))3)))))))))3)))))))))1
            *Prob. ID*Analysis*Design  *Program.*Sys.Tst.*Accept. *Delivery*


                        ▾              ▾                  ▾              ▾

            *         *              *                  *              *
MEDIUM      /)))))))))3)))))))))))))))3)))))))))))))))))))3)))))))))))))))1
            *Prob. ID*Anal./Design *Program./Test   *Accep./Deliv.*

                                  ▾                        ▾

            *                     *                        *
SMALL       /)))))))))0)))))))))))))3)))))))))))))))0)))))))))1
            *Prob. ID*Anal./Desi.*Program/Test *Acc./Del.*
            R         R           R              R         R


          LESS )))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))))Q▸ MORE
                     DEGREE OF PROJECT MANAGEMENT REQUIRED
```

▾ = Stage Exit occurs at this point.

[1] Size is used as a guide to help determine the appropriate degree of project management, and which stages may be combined for a given effort. Within this context, size is a combination of level of effort required (all activities) and complexity of the modification. Attributes of complexity include technology, team skills, interfaces, and level of understanding of requirements.
Other factors that can influence tailoring include risk, visibility, and business impact.

Exhibit 10.0-3. Process Model Metrics for Software Maintenance

	Problem Identification Stage	Analysis Stage	Design Stage	Programming Stage	System Test Stage	Acceptance Stage	Delivery Stage
Factors	Correctness Maintainability	Flexibility Traceability Usability Reusability Maintainability Comprehensibility	Flexibility Traceability Reusability Testability Maintainability Comprehensibility Reliability	Flexibility Traceability Maintainability Comprehensibility Reliability	Flexibility Traceability Verifiability Testability Interoperability Comprehensibility Reliability	Flexibility Traceability Interoperability Testability Comprehensibility Reliability	Completeness Reliability
Metrics	No. of omissions on Modification Request (MR) No. of MR submittals No. of duplicate MRs Time expended for problem validation	Requirement changes Documentation error rates Effort per function area (e.g., SQA) Elapsed time (schedule) Error rates, by priority and type	S/W complexity Design changes Effort per function area Elapsed time Test plans and procedure changes Error rates, by priority and type Number of lines of code, added, deleted, modified, tested	Volume/ functionality (function points or lines of code) Error rates, by priority and type	Error rates, by priority and type Generated Corrected	Error rates, by priority and type Generated Corrected	Documentation changes (i.e. version description documents, training manuals, operation guidelines)

Note: The above level of metrics is a goal. Each organization responsible for software maintenance activities should establish an individual plan to achieve this level over time.

Bibliography: The following materials were used in the preparation of the Software Maintenance chapter.

1. DeMarco, Tom, *Controlling Software Projects*, New York, 1989.

2. Frame, Davidson J., *Managing Projects in Organizations*, San Francisco, 1987.

3. Frame, Davidson J., *The New Project Management*, San Francisco, 1994.

4. Page-Jones, Meilir, *Practical Project Management*, New York, 1985.

5. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard for Software Maintenance*, IEEE Std 1219-1992, New York, 1993.

6. U.S. Department of Commerce/National Bureau of Standards, *Guideline on Software Maintenance*, Springfield, Virginia, 1984.

Stage: **10.1**
Problem/Modification Identification Stage

Responsibility: Maintenance Team

Description: In this stage, software changes are identified, classified, and assigned an initial priority ranking. Each request for a software modification (i.e., Modification Request) is evaluated to determine its classification and handling priority. The classification should be identified from the following types of maintenance.

- Corrective - Change to a software product after delivery to correct defects.
- Adaptive - Change to a software product after delivery to keep it functioning properly in a changed or changing environment.
- Emergency - Unscheduled corrective maintenance required to keep a system operational.

The need for software modifications can be driven by any number of factors, including:

- Report of system malfunction
- Mandatory changes required by new or changed federal or state law
- New requirements to support business needs
- Major enhancement or redesign to improve functionality or replace an obsolete system component
- Operational system upgrades and new versions of resident software (e.g., COBOL, CICS, Oracle)

These factors should be considered when assigning a priority to the modification request.

Exhibit 10.1-1 (provided at the end of this section) summarizes the input, process, control, and output for the Problem/Modification Identification Stage of software maintenance.

Input: Input to the Problem/Modification Identification Stage of software maintenance is one or more Modification Requests.

Process: If a modification to the software is required, the following activities must occur within the maintenance process:

- Assign an identification number
- Classify the type of maintenance
- Analyze the modification to determine whether to accept, reject, or further evaluate
- Prioritize the modification according to the following categories:
  - Emergency (follow emergency change procedure and integrate into the next scheduled release or block of modifications)
  - Mandatory (e.g., legal, safety, payroll)
  - Required (has associated benefits; e.g., productivity gains, new business drivers)
  - Nice to have (lower priority)

Control: Modification Requests and process determinations are uniquely identified and entered into the Project File.

Work Products: The output of this stage is the validated Modification Request and the following process determinations. Place a copy of all work products in the Project File.

- Statement of the problem or new requirement
- Problem or requirement evaluation
- Classification of the type of maintenance required
- Initial priority
- Verification data (for corrective modifications)
- Initial estimate of resources required

Review Process: Conduct peer review(s) as appropriate.

Exhibit 10.1-1. Problem/Modification Identification Stage

```
                 Uniquely identify Modification Request (MR)
                 Enter MR into Project File
                           T
                           *
                           ▾
                +)))))))))))))))))))))),
                * Problem/Modification *        Validated MR
MR  S))))▸      * Identification       *  S)))▸ Process determinations
                .)))))))))))))))))))))-
                           T
                           *

                    Metrics/Measure
                    Peer Review
```

Stage: **10.2**
Analysis Stage

Responsibility: Project Team and System Owner

Description: During the Analysis Stage, the Project File information, the Modification Request(s) validated in the Problem/Modification Identification Stage, and the system and project documentation are used to study the feasibility and scope of the modification, and to develop a preliminary Project Plan for design, test, and delivery.

If the documentation is not available or is insufficient and the source code is the only reliable representation of the software system, reverse engineering is recommended to ensure the overall integrity of the system. In those cases where long-lived systems have overgrown the initial base system and have poorly updated documentation, reverse engineering may be required and would evolve through the following steps:

For a smaller scope, or for local analysis on a unit level:

- Dissection of source code into formal units
- Semantic description of formal units and declaration of functional units
- Creation of input/output schematics of units

For a larger scope, or for global analysis on a system level:

- Declaration and semantic description of linear flows
- Declaration and semantic description of system applications (functions grouped)
- Creation of anatomy of the system (system architecture)

Modifications of a similar nature (i.e., affecting the same program(s)) should be grouped together whenever possible, and packaged into releases that are managed as projects. A release cycle should be established and published.

Exhibit 10.2-1 (provided at the end of this section) summarizes the input, process, control, and output for the Analysis Stage.

Input: Input to the Analysis Stage of software maintenance includes the following:

- Validated Modification Request
- Initial resource estimate and associated information
- Project and system documentation, if available

Process: Preliminary analysis activities include the following:

- Make a preliminary estimate of the modification size/magnitude
- Assess the impact of the modification
- Assign the Modification Request to a block of modifications scheduled for implementation
- Coordinate the modifications with other ongoing maintenance tasks

Once modifications are agreed to, grouped if appropriate, and packaged, analysis progresses and includes the following:

- Define firm requirements for the modification
- Identify elements of the modification
- Identify safety and security issues
- Devise a test and implementation strategy

In identifying the elements of the modification (creating the preliminary modification list), examine all work products (e.g., software, specifications, data bases, and documentation) that are affected. Each work product is identified, and generated, if necessary, specifying the portion of the product to be modified, the interfaces affected, the user-noticeable changes expected, the relative degree and kind of experience required to make changes, and the estimated time to complete the modification.

The test strategy is based on input from the previous activity identifying the elements of modification. Requirements for at least three levels of testing, including individual unit tests, integration tests, and user-oriented functional tests are defined. Regression test requirements associated with each of these levels of testing are identified as well. The test cases to be used for testing to establish the test baseline are revalidated.

Control: Control of the Analysis Stage activities includes the following:

- Retrieval of the relevant version of project and system documentation from the configuration control function of the organization.
- Review of the proposed changes and engineering analysis to assess technical and economic feasibility, and correctness.
- Identification of safety and security issues.
- Consideration of the integration of the proposed change within the existing software.
- Verification that all appropriate analysis and project documentation is updated and properly controlled.
- Verification that the test function of the organization is providing a strategy for testing the change(s), and that the change schedule can support the proposed test strategy.
- Review of resource estimates and schedules; verification of accuracy.
- Technical review to select the problem reports and proposed enhancements to be implemented in the new release.

Work Products: The output of the Analysis Stage includes the following:

- Feasibility report for modification requests
- Detailed analysis report
- Updated requirements (including traceability list)
- Test strategy
- Project Plan

A written assessment, generally called a Feasibility Report, is prepared and contains the following:

- Short and long term costs
- The value of the benefit of making the modification (usually provided by the system owner)
- Solution approach, including prototyping if applicable

Work Products, continued:

- Safety and security implications
- Human factors

A project plan states how the design, implementation, testing, and delivery of the modification is to be accomplished with a minimal impact to current users.

Review Processes: Conduct structured walkthrough(s), In-Stage Assessment(s), and a Stage Exit.

At the end of the Analysis Stage, a risk analysis is performed. Using the output of the Analysis Stage, the preliminary resource estimate is revised, and a decision is made on whether to proceed to the Design Stage.

Exhibit 10.2-1. Analysis Stage

```
                                Conduct technical review
                                Verify that documentation is updated
                                Verify test strategy
                                Identify safety and security issues
                                            T
                                            *
                                            ▾              Feasibility report
Validated Mod. Request         +)))))))))))),        Detailed analysis report
Project/system document S))))▸ *  Analysis  * S)))▸  Updated requirements
Project File information       .)))))))))))))-        Modification list
                                            T              Test strategy
                                            *              Project Plan
                                            R
                                     Metrics/measures
                                     Structured walkthrough
                                     In-Stage Assessment
                                     Stage Exit
```

Stage: **10.3**
Design Stage

Responsibility: Project Team

Description: In the Design Stage, all current system and project documentation, existing software and data bases, and the output of the Analysis Stage are used to design the modification to the system. *Exhibit 10.3-1* (provided at the end of this section) summarizes the input, process, and output for the Design Stage of software maintenance.

Input: Input to the Design Stage of software maintenance includes the following:

- Analysis Stage output, including:
  - Detailed analysis
  - Updated statement of requirements
  - Preliminary modification list (identification of affected elements)
  - Test strategy
  - Project Plan
- System and project documentation
- Existing source code, comments, and data bases

Process: The process steps for the Design Stage include the following:

- Identify selected software modules
- Modify software module documentation (e.g., data and control flow diagrams, schematics)
- Create test cases for the new design, including safety and security issues
- Identify/create regression tests
- Identify documentation (system/user) update requirements
- Update modification list
- Document any known constraints that influence the design, and any risks that have been identified. Where possible, actions, taken or recommended, that mitigate risk should also be documented.

Control: The following control activities are performed during the Design Stage of a modification.

- Conduct structured walkthrough(s) of the design
- Verify that the new design/requirement is documented as an authorized change
- Verify the inclusion of new design material, including safety and security issues
- Verify that the appropriate test documentation has been updated
- Complete the traceability of the requirements to the design

Work Products: The output of the Design Stage of software maintenance includes the following:

- Revised modification list
- Updated design baseline
- Updated test plans
- Revised detailed analysis
- Verified requirements
- Updated Project Plan
- Documented constraints and risks

Review Process: Conduct structured walkthrough(s), In-Stage Assessment(s), and a Stage Exit.

Exhibit 10.3-1. Design Stage

```
                          Conduct structured walkthrough(s)
                          Verify that design is documented
                          Ensure traceability of requirements to design
                                       T
                                       *
                                       ▾              Revised modification list
System/project document         +)))))))))))),       Updated design baseline
Analysis Stage output   S))))▸  *   Design   * S)))▸ Updated test plans
Source code, data base          .)))))))))))))-       Validated requirements
                                       T              Updated Project Plan
                                       *              Constraints and risks
                                       R
                               Metrics/measures
                               Structured walkthrough(s)
                               In-Stage Assessment(s)
                               Stage Exit
```

Stage: **10.4**
Programming Stage

Responsibility: Project Team

Description: In the Programming Stage, the results of the Design Stage, the current source code, the project and system documentation, (i.e., the entire system as updated by the prior stages) is used to drive the programming effort. *Exhibit 10.4-1* (provided at the end of the section) summarizes the input, control, and output for the Programming stage.

Input: Input to the Programming Stage of software maintenance includes the following:

- Results of the Design Stage
- Current source code, comments, and data bases
- Project and system documentation

Process: The Programming Stage includes the following tasks, which may be conducted in an incremental, iterative approach:

- Coding and unit testing
- Integration
- Revisit project risk
- Test readiness review

Coding and Unit Testing: Implement the change into the code and perform unit testing. Other quality assurance and verification and validation processes may be required for safety-related code. The Quality Assurance Team can help with specific issues.

Integration: After the modifications are coded and unit tested, or at appropriate intervals during coding, the modified software is integrated with the system, and integration and regression tests are refined and performed. All effects (e.g., functional, performance, usability, safety) of the modification on the existing system are assessed and noted. A return to the coding and unit testing tasks is made to remove any unacceptable impacts.

Risk Analysis and Review: Risk analysis and review are performed periodically during the Programming Stage rather than at the end, as in the Design and Analysis Stages. Metrics/measurement data should be used to quantify risk analysis.

Test Readiness Review: To assess the team's preparedness to enter system testing, a Test Readiness Review is conducted. This is a self assessment to determine if items including code, documentation, libraries, hardware, telecommunication lines, and schedules are ready for system test to begin on the scheduled date.

Control: Control of the Programming Stage includes the following activities:

- Conduct structured walkthroughs of the code
- Ensure that unit and integration testing are performed and documented in the Project File
- Ensure that test documentation (e.g., test plans, test cases, and test procedures) are either updated or created
- Identify, document, and resolve any risks exposed during software and test readiness reviews
- Verify that the new software is placed under software configuration management control
- Verify that the training and technical documentation have been updated
- Verify the traceability of the design to the code

Work Products: The output of the Programming Stage includes the following:

- Updated software
- Updated design documentation
- Updated test documentation
- Updated user documentation
- Updated training material
- Statement of risk and impact to users
- Test Readiness Review report

Review Process: Conduct structured walkthrough(s), In-Stage Assessment(s), and a Stage Exit.

Exhibit 10.4-1. Programming Stage

```
                              Conduct structured walkthrough
                              Ensure testing performed and documented
                              Verify:
                                New software placed under configuration management
                                Documentation has been updated
                                Traceability of design to code
                                           T
                                           *
Results of Design Stage                    ▾                Updated:
Source code                     +)))))))))))),               Software
Project documentation   S))))▸ *Programming * S)))▸          Design documents
System documentation            .)))))))))))-                Test documents
                                           T                 User documents
                                           *                 Training material
                                           R                 Project Plan
                                                            Test Readiness Rev. rpt.
                                 Metrics/measures
                                 Structured walkthrough(s)
                                 In-Stage Assessment(s)
                                 Stage Exit
```

Stage: **10.5**
System Test Stage

Responsibility: Independent Tester(s) or Project Team

Description: System testing is performed on the modified system. Regression testing is a part of system testing and is performed to validate that the modified code does not introduce faults that did not exist prior to the maintenance activity. *Exhibit 10.5-1* (provided at the end of the section) summarizes the input, process, control, and output for the System Test Stage.

Input: Input to the System Test Stage of software maintenance includes the following:

- Test Readiness Review report
- Documentation, which includes:
  - System test plans(s)
  - System test cases
  - System test procedures
  - User manuals
  - Design
- Updated system
- Updated Project Plan

Process: System tests are conducted on a fully integrated system. Testing shall include the performance of:

- System functional test
- Interface testing
- Regression testing
- Test readiness review to assess preparedness for acceptance testing

Results of tests conducted prior to the test readiness review should not be used as part of the system test report to substantiate requirements at the system level. This is necessary to assure that the test organization does not consider that testing all parts (one at a time) of the system constitutes a "system test."

Control: System tests should be conducted by an independent party for maximum results. Prior to the completion of system testing, the test function is responsible for reporting the status of the activities that had been established in the test plan for

Control, continued: satisfactory completion of system testing. The status is reported to the appropriate reviewers prior to proceeding to acceptance testing. Software code listings, Modification Requests, and test documentation are placed under configuration management. The system owner shall participate in the review to ascertain that the maintenance release is ready to begin acceptance testing.

Work Products: The output for the System Test Stage of software maintenance includes the following:

- Tested and fully integrated system
- Test report
- Test Readiness Review report
- Updated Project Plan

Review Process: Conduct structured walkthrough(s), In-Stage Assessment(s), and a Stage Exit.

Exhibit 10.5-1. System Test Stage

```
                                 Place under configuration management:
                                  Software code and listings
                                  Modification Requests
                                  Test documentation
                                            T
                                            *
Updated software                            ▾
Documentation                      +)))))))))))),        Tested, integrated system
Test Read. Review rept. S))))▸ *System Test * S)))▸  Test reports
Updated system                     .)))))))))))-        Test Read. Review report
                                            T             Updated Project Plan
                                            *
                                            R

                                   Metrics/measures
                                   Structured walkthrough(s)
                                   In-Stage Assessment(s)
                                   Stage Exit
```

Stage: **10.6**
Acceptance Stage

Responsibility: System Owner/User or other designated individuals

Description: Acceptance tests are conducted on a fully integrated system. Acceptance tests are performed by either the system owner, the user of the modification package, or a third party designated by the system owner. An acceptance test is conducted on the modified system, with software that is under software configuration management in accordance with the application's Software Configuration Management Plan. *Exhibit 10.6-1* (provided at the end of this section) summarizes the input, process, control, and output for the Acceptance Stage.

Input: Input for the Acceptance Stage of software maintenance includes the following:

- Test Readiness Review report
- Fully integrated system
- Acceptance Test Plan
- Acceptance test cases
- Acceptance test procedures

Process: The following steps form the process for acceptance testing:

- Perform acceptance tests at the functional level

- Perform interoperability testing (to validate the functionality of any input and output interfaces)

- Perform regression testing

- Conduct a Functional Configuration Audit (FCA)

The purpose of a FCA is to verify that all requirements specified and agreed to have been met. The FCA compares the system's software elements (programs/modules) to the software requirements documented in the current version of the Software Requirements Specification to assure that the modification addresses all, and only, those requirements. The results of the FCA should be documented, identifying all discrepancies found, and the plans for their resolution.

Control: Control of acceptance tests includes the following:

- Execute acceptance tests
- Report results for the Functional Configuration Audit conducted to ensure that all of the functionality that has been agreed to is in fact present in the system
- Establish the new system baseline
- Place the acceptance test documentation under software configuration management control

Work Products: The output of the Acceptance Stage includes the following:

- New system baseline
- Functional Configuration Audit Report
- Acceptance Test Report
- Updated Project Plan

Review Processes: Conduct structured walkthrough(s), In-Stage Assessment(s), and a Stage Exit.

Exhibit 10.6-1. Acceptance Stage

```
                                Execute acceptance tests
                                Report test results
                                Conduct functional audit
                                Establish new baseline
                                  Acceptance test documentation under configuration
      management
                                          T
                                          *
    Test Readiness Review report          ▾
    Fully integrated system        +)))))))))))),        New system baseline
    Acceptance Test Plan    S))))▸ *Accept. Test* S)))▸  Functional configuration
    Acceptance test cases          .)))))))))))-          audit report
    Acceptance test procedures            T              Acceptance test report
                                          *              Updated Project Plan
                                          R

                                  Metrics/measures
                                  Structured walkthrough(s)
                                  In-Stage Assessment(s)
                                  Stage Exit
```

Stage: **10.7**
Delivery Stage

Responsibility: Project Team

Description: This stage describes the requirements for the delivery of a modified software system. *Exhibit 10.7-1* (provided at the end of the section) summarizes the input, process, control, and output for the Delivery Stage.

Input: Input to the Delivery Stage of software maintenance is the fully tested version of the system as represented in the new baseline.

Process: The tasks for delivery of a modified system include the following:

- Conduct a Physical Configuration Audit (PCA).
- Notify the user community.
- Develop an archival version of the system for backup.
- Perform installation and training at the user facility.

The purpose of a PCA is to verify that the software associated with the modification and its documentation are internally consistent and are ready for delivery. The PCA compares the software components (programs/modules) with its supporting documentation to assure that the documentation to be delivered correctly describes the system components. All discrepancies noted during the PCA, along with plans for their resolution should be documented.

Control: Control for the Delivery Stage includes the following:

- Arrange and document a Physical Configuration Audit
- Provide access to system materials for users, including replication and distribution
- Complete the version description document
- Place under software configuration management control

Work Products: The output of the Delivery Stage includes the following:

- Physical Configuration Audit report
- Version Description Document (VDD).

The VDD contains information pertinent to the version or release of the system that is being delivered. Information provided includes system name, date delivered, version number, release number, brief description of functionality delivered in the modification, and prerequisite hardware and software with its associated version and release number. The current VDD is placed together with VDDs from previous versions/releases to form a complete chronology of the system from its initial implementation or Version 1,
Release 1.

Review Process: Conduct structured walkthrough(s) and a Stage Exit.

Exhibit 10.7-1. Delivery Stage

```
                        Arrange physical configuration audit
                        Complete version description document
                                        T
                                        *
                                        ▾
                                 +)))))))))))),        Physical configuration
Tested/accepted system  S))))▸ *  Delivery  * S)))▸   audit report
                                 .)))))))))))-        Version description doc.
                                        T
                                        *
                                        R

                               Metrics/measures
                               Structured walkthrough(s)
                               Stage Exit
```

Software Engineering Methodology

Appendix A
Glossary

Acceptance criteria
The criteria that a software component, product, or system must satisfy in order to be accepted by the system owner or other authorized acceptance authority.

Acceptance process
The process used to verify that a new or modified software product is fully operational and meets the system owner's requirements. Successful completion of the acceptance process results in the formal transfer of the software product responsibilities from development to maintenance personnel.

Acceptance testing
Formal testing conducted to determine whether or not a software product or system satisfies its acceptance criteria and to enable the system owner to determine whether or not to accept the product or system.

Activity
A major unit of work to be completed in achieving the objectives of a software project. An activity incorporates a set of tasks to be completed, consumes resources, and results in work products. An activity may contain other activities in a hierarchical manner. All project activities should be described in the Project Plan.

Algorithm
A finite set of well-defined rules for the solution to a problem in a finite number of steps. Any sequence of operations for performing a specific task.

Anomaly
Anything observed in the operation or documentation of software that deviates from expectations based on previously verified software products or documents.

Application
Software products designed to fulfill specific needs.

Assumption
A condition that is taken to be true without proof or demonstration.

Audit
An independent examination of a work product to assess compliance with specifications, standards, quality or security requirements, contractual agreements, or other predetermined criteria.

Baseline
A set of configuration items (software components and documents) that has been formally reviewed and agreed upon, that serves as the basis for further development, and that can be changed only through formal change control procedures.

Baselined requirements
The set of project requirements that have been approved and signed off by the system owner during the Requirements Definition Stage. The software product design is based on these requirements. The baselined requirements are placed under configuration control.

Code
Computer instructions and data definitions expressed in a programming language or in a form that is output by an assembler, compiler, or other translator.

Code generator
A software tool that accepts as input the requirements or design for a computer program and produces source code that implements the requirements or design.

Code review
A meeting at which software code is presented to project personnel, managers, users, or other functional areas for review, comment, or approval.

Component
One of the parts that make up a system. A component may be hardware, software, or firmware and may be subdivided into other components.

Computer-Aided Software Engineering (CASE)
The use of computers to aid in the software engineering process. May include the application of software tools for software design, requirements tracing, code production, testing, document generation, and other software engineering activities.

Configuration control
An element of configuration management consisting of the evaluation, coordination, approval/disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.

Configuration Control Board
A group of people responsible for evaluating and approving/disapproving proposed changes to configuration items, and for ensuring implementation of approved changes.

Configuration item
An aggregate of hardware or software components that are designated for configuration management and treated as a single entity in the configuration management process.

Configuration management See Software configuration management

Constraint
A restriction, limit, or regulation that limits a given course of action or inaction.

Cost estimate
A formal estimate of the cost to develop and support a project. Estimates should reflect all activities such as design, development, coding, distribution, service, and support of the product; staffing; training and travel expenses; subcontractor activities; contingencies; and cost for external services (e.g., technical documentation production and Quality Assurance audits and reviews).

Deliverable
A work product that is identified in the Project Plan and is formally delivered to the system owner and other project stakeholders for review and approval.

Dependency
A relationship of one task to another where the start or end date of the second task is related to the start or end date of the first task.

Design
The process of defining the architecture, components, interfaces, and other characteristics of a software product or component.

Design specification
A document that describes the design of a software component, product, or system. Typical contents include architecture, control logic, data structures, input/output formats, interface descriptions, and algorithms.

Feasibility
The degree to which the requirements, design, or plans for a software product or system can be implemented under existing constraints.

Functional area
Any formally organized group involved in the development and maintenance of software or the support of development and maintenance efforts, or other group whose input is required to successfully implement a software project. Examples of functional areas include software engineering services, technical writing, quality assurance, security, and telecommunications.

Functional Design Stage
The period of time in the software lifecycle during which the designs for architecture, software components, interfaces, and data are created, documented, and verified to satisfy project requirements.

Functional requirement
A requirement that specifies a function that a software component, product, or system must be able to perform.

Functional Test Plan
A plan for testing each function across one or more units. The plan describes how the functional testing occurs and the test procedure/test cases that will be used. The plan includes procedures for creating the test environment that allows all functions to be executed; the entry and exit criteria for starting and ending the function testing period; and the schedule followed for starting and ending each test.

Functional test procedures
Procedures for each function or combination of functions to be tested. Procedures fully describe how the function is tested. Expected output from each test procedure is identified to compare the planned output to actual output.

Functional testing
Testing conducted to evaluate the compliance of a software product with specified functional requirements. Testing that focuses on the outputs generated in response to selected inputs and execution conditions.

Hardware
Physical computer and other equipment used to process, store, or transmit computer programs or data.

Hierarchy
A structure in which components are ranked into levels of subordination.

Implementation requirements
A requirement that supports the development and maintenance concepts and approaches in the areas of operating environment, conversion, installation, training, and documentation.

Incremental development
A software development technique in which requirements definition, design, implementation, and testing occur in an overlapping, iterative (rather than sequential) manner, resulting in incremental completion of the overall software product.

Information Engineering
A development methodology where models are created to improve the users' ability to understand and define the functions and flow of information within their organization. A business model is developed to identify the key areas of interest for the business, the tasks required for each area, and the activities that make up each task. The business model prioritizes and identifies top management goals and then establishes the information needs necessary to reach those goals. A data model is developed to describe the data and the relationships among data. The data model further divides the business model into user-defined relationships (e.g., entity relationship model).

Inspection
A static analysis technique that relies on visual examination of development products to detect errors, violations of development standards, and other problems. Code inspection and design inspection are two types.

Integration testing
An orderly progression of testing in which software components are combined and tested to evaluate the interaction between them.

Integrity
The degree to which a software component, product, or system prevents unauthorized access to, or modification of, computer programs or data.

Interactive analysis and design
A development methodology that uses facilitated team techniques, such as Joint Application Development or Rapid Application Development, to rapidly develop project requirements that reflect the users' needs in terminology that the users understand. Group facilitation techniques are especially important when several user organizations have unique project requirements that are specific to their mission and goals.

Interface requirement
A requirement that specifies an external item with which a software product or system must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction.

Interface testing
Testing conducted to evaluate whether software components pass data and control correctly to one another.

Key process area
Software engineering processes identified by the Software Engineering Institute Capability Maturity Model where a project team should focus its efforts to achieve consistently high quality software products.

Lifecycle See Software lifecycle.

Maintenance
The process of supporting a software product or system after delivery to maintain operational status, correct faults, improve performance or other attributes, or adapt to a changed environment.

Menu-driven
Pertaining to a system or mode of operation in which the users direct the software through menu selections.

Methodology
A collection of methods, procedures, and standards that defines an integrated synthesis of engineering approaches to the development of a work product.

Milestone
A scheduled event for which an individual or team is accountable and that is used to measure progress.

Module
A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. A logically separable part of a program.

Module testing
Testing of individual software modules or groups of related modules to verify the implementation of the design.

Performance requirement
A requirement that imposes conditions on a functional requirement (e.g., a requirement that specifies the speed, accuracy, or memory usage with which a given function must be performed).

Planning Stage
The initial stage in the software lifecycle during which the system owner/users' needs and expectations are identified, the feasibility of the project is determined, and the Project Plan is developed.

Platform
A specific computer and operating system on which a software product is developed or operated.

Portability
The ease with which a software component, product, or system can be transferred from one hardware or software environment to another.

Procedure
A written description of a course of action to be taken to perform a given task.

Process
An ordered set of steps performed for a given purpose. Processes define or control the development of the project work products. The use of processes will ensure a consistent methodology across all platforms in producing the lifecycle deliverables.

Product See Work product.

Programmers Reference Manual
A work product deliverable that provides information necessary to maintain or modify software for a given computer system. Typically described are the equipment configuration, operational characteristics, programming features, input/output features, and compilation or assembly features of the computer system.

Programming Stage
The period of time in the software lifecycle during which a software product is created from the design specifications and testing is performed on the individual software units.

Project
An undertaking requiring concerted effort that is focused on developing or maintaining a specific software product or system. A project has its own funding, cost accounting, and delivery schedule.

Project File
A central repository of material pertinent to a project. Contents typically include all work products, memos, plans, technical reports, and related items.

Project lifecycle
The software lifecycle selected for the project and approved by the system owner and other project stakeholder(s).

Project manager
The individual with total business responsibility for all software activities of a project. The project manager directs, controls, administers, and regulates a project.

Project Plan
A document that describes the technical and management approach to be followed for a project. The plan typically describes the work to be done, the resources required, the methods to be used, the procedures to be followed, the schedules to be met, and the way the project will be organized. The plan includes a list of deliverables, actions required, and other key events needed to accomplish the project.

Project Team
The project manager, analysts, programmers, and other staff assigned as the core group for a project. The project team may include representatives of the other functional areas (e.g., technical writer and telecommunications expert) responsible for contributing to the development, installation, and maintenance of the software product.

Project Test Plan
Defines all test activities required to assure that the software product will perform satisfactorily for all users. As a minimum, the plan should include descriptions for unit testing, integration testing, system testing, and acceptance testing.

Prototyping
A technique for developing and testing a preliminary version of the software product (either as a whole or in modular units) in order to emulate functionality without such encumbering features as error handling, help messages, security controls, and other utilities that are not part of the design logic. This allows the project team to test the overall logic and workability of required functions and provides a model by which the project team and users can jointly determine if the software requirements meet the intended objectives. Prototyping is often used in conjunction with interactive analysis and design techniques.

Pseudocode
A combination of programming language constructs and natural language used to express a computer program design.

Rapid Prototyping
A type of prototyping in which emphasis is placed on developing prototypes earlier in the development process to permit early feedback and analysis in support of the development process.

Reference
A document(s) or other material that is useful in understanding more about an activity.

Regression testing
Selective retesting of a software component to verify that modifications have not caused unintended effects and that the software component still complies with its specified requirements.

Reliability
The ability of a software component to perform its required functions under stated conditions for a specified period of time.

Requirement
A condition or capability needed by a system owner/user to solve a problem or achieve an objective. A condition or capability that must be met or possessed by the software product to satisfy a contract, standard, specification, or other formally imposed documents.

Requirements analysis
The process of studying system owner/user(s) needs to arrive at a definition of system, hardware, or software requirements.

Requirements Definition Stage
The period of time in the software lifecycle during which the requirements for a software product are defined and documented.

Requirements management
A Software Engineering Institute Capability Maturity Model key process area designed to establish a common understanding between the system owner/user and the project team regarding the system owner/users' software requirements. This understanding forms the basis for estimating, planning, performing, and tracking the project's activities throughout the lifecycle.

Requirements Specification
A work product deliverable that specifies the manual and automated requirements for a software product in nontechnical language that the system owner/users can understand. Typically included are functional requirements, performance requirements, and interface requirements. Describes in detail what will be delivered in the product release.

Retirement
Permanent removal of a system or software product from its operational environment.

Reusability
The degree to which a software module or other work product can be used in more than one computer program or software system.

Reverse engineering
A development methodology in which the software development process is performed in reverse. The technique involves the examination of an existing software product that has characteristics that are similar to the desired product. Using the existing code as a guide, the requirements for the product are defined, analyzed, and abstracted all the way back to specifications. Any required code changes can be made based on a specification-like format. Ideally, the specifications would be edited and passed to a code generator that would trigger automatic documentation and revisions. Once testing is complete, the revised code is placed into production.

Risk
The possibility of suffering loss.

Risk management
An approach to problem analysis that is used to identify, analyze, prioritize, and control risks.

Software
Computer programs, procedures, and associated documentation and data pertaining to the operation of a software product or system.

Software configuration item
An aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

Software configuration management
(1) A discipline that effectively controls and manages all modifications to a software component, product, or system. Technical and administrative processes and tools are used to identify and document the functional and physical characteristics of the configuration items, manage and track changes to those items, record and report change processing and implementation status, and verify compliance with specified requirements.
(2) A Software Engineering Institute Capability Maturity Model key process area designed to establish and maintain the integrity of the software work products throughout the project's lifecycle.

Software Engineering Methodology
The Departmental methodology that identifies the processes, activities, tasks, management responsibilities, and work products that are required for each software development and maintenance project. Deviations from the methodology require the approval of all parties who have approval rights on the project. A key objective of the methodology is to provide measurable, repeatable processes to assure that project development and maintenance methodologies are consistent throughout the Departmental information systems environment.

Software lifecycle
The period of time that begins when a software product is conceived and ends when the software is retired. A network of stages and processes that function together to guide the development and maintenance of software products. Each process produces a set of deliverables as it moves through the lifecycle.

Software Process Planning
A Software Engineering Institute Capability Maturity Model key process area designed to establish reasonable plans for performing software engineering and for managing the software project.

Software Project Tracking and Oversight
A Software Engineering Institute Capability Maturity Model key process area designed to provide adequate visibility into actual project progress so that management can take effective actions when the project's performance deviates significantly from the plans.

Software Quality Assurance
A Software Engineering Institute Capability Maturity Model key process area designed to provide management with appropriate visibility into the software engineering processes being used by the project team and the work products being built.

Software System
A software product and the documentation, hardware, and telecommunications needed to implement and operate the product and accomplish a specific function or set of functions.

Specification
A document that specifies in a complete, precise, verifiable manner the requirements, design, behavior, or other characteristics of a software component, product, or system.

Spiral development model
A software development process in which the constituent activities, typically requirements analysis, design, coding, integration, and testing are performed iteratively until the software product is complete.

Stage
A partition of the software lifecycle that reduces a project to manageable size and represents a meaningful and measurable set of related tasks that are performed to obtain specific work products.

Stakeholder
The DOE individual(s) with decision-making authority over a project or group of projects.

Standard
Mandatory requirements employed and enforced to prescribe a disciplined, uniform approach to software development and maintenance.

Structured analysis
An analysis technique that uses a graphical language to build models of software products or systems. The four basic features in structured analysis are data flow diagrams, data dictionaries, procedure logic representations, and data store structuring techniques.

System
A collection of hardware, software, firmware, and documentation components organized to accomplish a specific function or set of functions.

System Design Document
A work product deliverable that describes the solution to the automation task as described by the requirements. Contains sufficient detail to provide necessary direction for writing the Program Specifications and allows developers maximum technical freedom.

System Design Stage
A stage in the lifecycle model during which the designs for the software product architecture, software components, interfaces, and data are refined and expanded to the extent that the design is sufficiently complete to be implemented.

System owner
The organizational unit that funds and has approval authority for the project. Typically, system owners are also system users.

System testing
Testing conducted on a complete, integrated software product or system to evaluate compliance with its specified requirements.

User
The general population of individuals who use a software product or system. User activities can include data entry; read only; add, change and delete capabilities; querying; and report generation.

Task
The smallest unit of work subject to management accountability. A task is a well-defined work assignment for one or more project team members. Related tasks are usually grouped to form activities. An task is the lowest level of work division typically included in the Project Plan and Work Breakdown Structure.

Test bed
An environment containing the hardware, instrumentation, simulators, software tools, and other support elements needed to conduct a test.

Test case
A set of test inputs, execution conditions, and expected results that are developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test criteria
The criteria that a software component or product must meet in order to pass a given test.

Test design
Documentation specifying the details of the test approach for a software feature or combination of software features and identifying the associated tests.

Test documentation
Documentation describing plans for, or results of, the testing of a software component or product. Documents typically include test case specifications, test incident reports, test logs, test plans, test procedures, and test reports.

Test item
A software item that is the object of testing.

Test log
A chronological record of all relevant details about the execution and results of a test.

Test phase
The period of time in the software lifecycle in which the components of a software product are evaluated and integrated, and the software product is evaluated to determine whether or not the requirements have been satisfied.

Test plan
A document specifying the scope, approach, resources, and schedule of intended testing activities. The plan identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

Test procedure
Detailed instructions for the setup, execution, and evaluation of the results for a given test case.

Test report
A document that describes the conduct and results of the testing carried out for a software component or product.

Testing
An activity in which a software component or product is executed under specified conditions, the results are observed and recorded, and an evaluation is made.

Traceability
The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor relationship to one another.

Interview technique
An technique for the identification, analysis, and documentation of the project requirements. The project team conducts a series of interviews with users to identify the users' perceived automated functional needs, analyzes the information gathered during the interviews, and develops the requirements.

Transaction analysis
A technique used to derive structured charts for a software product that will process transactions. Transaction analysis is used to divide complex data flow diagrams into smaller, simpler data flow diagrams--one for each transaction that the product or system will process. Structure charts are developed from the simple data flow diagrams. The individual structure charts for the separate transactions are then combined to form one large structure chart that is very flexible and can accommodate user changes.

Unit
A separately testable element specified in the design of a computer software component. A software component that is not subdivided into other components.

Unit testing
Testing of individual hardware or software units or groups of related units. The isolated testing of each flowpath of code with each unit. The expected output from the execution of the flowpath should be identified to allow comparisons of the planned output against the actual output.

Usability
The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a software product.

User interface
An interface that enables information to be passed between a user and hardware or software components of a computer system.

User manual
A document that presents the information necessary to use a software product to obtain desired results. Typically described are product or component capabilities, limitations, options, permitted inputs, expected outputs, possible error messages, and special instructions.

Validation
The process of evaluating software at the end of the software development process to assure compliance with established software and system requirements.

Verification
The process of evaluating a software product to determine whether or not the work products of a stage of the software lifecycle fulfill the requirements established during the previous stage.

Walkthrough
An analysis technique in which a team of subject matter experts review a segment of code or documentation, ask questions, and make comments about possible errors, violation of development standards, and other problems.

Work product
Any tangible item that results from a project function, activity, or task. Examples of work products include process descriptions, plans, procedures, computer programs, and associated documentation, which may or may not be intended for delivery to the system owner and other project stakeholders.

Bibliography: The following materials were used in the preparation of the glossary.

1. Software Engineering Institute, Software Process Maturity Questionnaire, Capability Maturity Model, version 1.1, Carnegie Mellon University, Pittsburgh, 1994.

2. The Institute of Electrical and Electronics Engineers, Inc., *IEEE Standard Glossary of Software Engineering Terminology*, IEEE Std 610.12-1990, The Institute of Electrical and Electronics Engineers, Inc., New York, 1990.

3. U.S. Department of Labor, Directorate of Information Resources Management, *Systems Engineering Concepts and Procedures Manual*, U.S. Department of Labor, 1988.

Software Engineering Methodology

Appendix B
List of Abbreviations

ABC	Analysis of Benefits and Costs
ACPPM	Assistant Computer Protection Program Manager
ANSI	American National Standards Institute
CASE	Computer-Aided Software Engineering
CICS	Customer Information Control System
CPP	Computer Protection Plan
CSSO	Computer Systems Security Officer
DOE	Department of Energy
DOS	Disk operating system
ESTSC	Energy Science and Technology Software Center
FCA	Functional Configuration Audit
IBM	International Business Machines
IEEE	The Institute of Electrical and Electronics Engineers, Inc.
IP	Internet protocol
ISA	In-Stage Assessment
LAN	Local area network
MASL	Microcomputer Applications System Library
MVS	Multiple virtual storage
PCA	Physical Configuration Audit
POC	Point of contact
SASREPS	Survey of Administrative Systems Reporting System
SEI	Software Engineering Institute (at Carnegie-Mellon)
SQA	Software Quality Assurance
SRIS	Systems Review Inventory System
Std	Standard
TCP	Transmission control protocol
VDD	Version Description Document
VM	Virtual machine
WAN	Wide area network

Software Engineering Methodology

Appendix C
Conducting Structured Walkthroughs

Purpose: This process guide describes how to conduct a structured walkthrough during the lifecycle stages of software engineering projects, regardless of hardware platform.

Reference: Yourdon, Edward, *Structured Walkthroughs*, New York, New York: Yourdon, Inc., 1978.

Organization: This process guide consists of the following sections:

- Overview
- Responsibilities Before the Walkthrough
- Responsibilities During the Walkthrough
- Responsibilities After the Walkthrough
- Additional Activities After the Walkthrough
- Follow-up Activities
- Structured Walkthroughs for Lifecycle Stages
- Structured Walkthroughs for Other Documents
- Structured Walkthrough Meeting Record (sample)
- Structured Walkthrough Management Summary Report (sample)

Description: A structured walkthrough is an organized procedure for a group of peers to review and discuss the technical aspects of software development and maintenance work products. The major objectives in a structured walkthrough are to find errors and to improve the quality of the product. Errors typically occur as omissions or contradictions, flaws in logic, or inconsistencies in the work product style (e.g., poorly stated requirements and inefficient code).

Structured walkthroughs should not be used to discuss solutions for the errors that are found. *The basic purpose of a walkthrough is error detection, not error correction.* When the walkthrough is finished, the author of the work product is responsible for taking the necessary actions to correct the errors. The author may hold private conversations with reviewers or conduct follow-up meetings to discuss potential solutions.

Structured walkthroughs should be conducted during all stages of the software lifecycle. Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants.
In some cases, it might be useful and expedient to include software users in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the

basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same.

General Information: Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of software development and maintenance work products, project management tools, and other types of documents (e.g., technical operating procedures). The walkthroughs should be scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage should determine the frequency of the walkthroughs.

Benefits: Structured walkthroughs provide the following benefits.

- Save time and money by finding and correcting errors earlier in the lifecycle.
- Provide value-added input from reviewers with different technical backgrounds, experience, and expertise.
- Validate and improve the related lifecycle work products.
- Keep the project team informed of the development or maintenance progress.
- Provide professional growth to participants by giving them an opportunity to look at different development or maintenance methodologies and approaches.

Participants: Each participant in the structured walkthrough process has a specific role.

The **author** of the work product is responsible for requesting the walkthrough when a meaningful portion of the product has been developed and is free from casual errors (e.g., spelling errors). The author attends the walkthrough as an observer and answers reviewers general questions. The author is not a reviewer.

The **presenter** usually develops the agenda for the walkthrough and presents the work product being reviewed. The presenter should be familiar with the work product and be a member of the project team.

Participants, continued:

The **moderator** facilitates the walkthrough session, ensures that the walkthrough agenda is followed, and encourages the participation of all reviewers. The moderator may also be the scribe.

The **reviewers** evaluate the work product to determine if it is technically accurate. The reviewers also assess whether the project guidelines or standards are being followed, the project requirements are met, and the product is properly prepared.

The **scribe** takes notes during the walkthrough. The scribe records the errors identified and any other technical comments, suggestions, and unresolved questions. The scribe should not be a reviewer.

Meeting Record:

The Structured Walkthrough Meeting Record worksheet is available to assist the reviewers with recording errors found prior to the walkthrough session, and for the scribe to record information discussed during the walkthrough. The worksheet is divided into two parts: Part 1 is used to record administrative meeting information; Part 2 is used to record reviewer comments, questions, and follow-up action items. A template of the worksheet is provided as an attachment to this procedure.

Implementation:

This procedure describes a formal structure for conducting walkthroughs. The formality and structure of the walkthrough sessions should be tailored to meet the needs of the development or maintenance team, and the purpose and scope of the work product.

Author's Responsibilities: The author of the work product is responsible for the following activities prior to the walkthrough session.

Step	Activity
1	Complete a meaningful segment of a work product. Avoid requesting a walkthrough on an incomplete segment or on a segment(s) that is too large to be adequately reviewed in less than 2 hours.
2	Proofread work product segment to eliminate nontechnical errors such as spelling or typographical mistakes. Nontechnical errors can distract reviewers from the technical aspects of the work product.
3	Notify the presenter that a completed segment of a work product is ready for a structured walkthrough. The author may discuss potential reviewers with the presenter.
4	Prepare any support materials (such as flow charts) to assist reviewers with their understanding of the entire work product and how the segment being reviewed fits into the entire product.
5	Provide the work product and all support materials to the presenter for advance distribution to the reviewers.
6	When the segment to be reviewed is finished, the author should be prepared to work on other segments of the work product (or other project tasks) while waiting for the walkthrough to occur.

Presenter's Responsibilities: The presenter is responsible for the following activities prior to the walkthrough session.

Step	Activity
1	Determine if the size of the work product segment is appropriate for one walkthrough session. The duration of a walkthrough session should not exceed 2 hours. If more time is necessary, the work product segment should be divided into smaller portions and each portion reviewed separately.
2	Select reviewers who are appropriate for the work product; such as systems analysts, programmers, technical writers, and testers. Reviewers should include people on and off the project. In some cases, the participation of software users may be considered desirable. If necessary, the presenter can discuss who should participate in the walkthrough with the manager of the project team.
3	Select the moderator and the scribe. Determine whether the scribe will be responsible for completing the Structured Walkthrough Management Summary Report.
4	Schedule the meeting date, time, and location. Notify all participants of these arrangements at least 2 days prior to the walkthrough.
5	Establish the agenda. Review the agenda and any important issues with the moderator.
6	Provide reviewers with copies of all materials to be reviewed at least 2 days prior to the walkthrough. The review package should include a blank copy of the Structured Walkthrough Meeting Record worksheet for optional use by reviewers.

Reviewers' Responsibilities: The reviewers are responsible for the following activities prior to the walkthrough session.

Step	Activity
1	Carefully review the materials provided by the presenter. Make a note about the amount of time spent reviewing the material. Give this information to the scribe at the beginning of the walkthrough session.
2	Identify technical errors. Insert comments and questions directly on the review materials or on part 2 of the Structured Walkthrough Meeting Record worksheet for easy reference during the walkthrough discussion.
3	Note directly on the review materials any nontechnical errors found during the review, such as spelling or typographical mistakes. While these errors are not discussed during the walkthrough, they should be provided to the author at the conclusion of the walkthrough.
4	Notify the presenter immediately if the reviewer will not be able to complete the review in time for the walkthrough session. An unprepared reviewer will hinder the walkthrough process. If enough time is available, the presenter can select a new reviewer.
5	Review the procedures for the structured walkthrough process. Each reviewer should be familiar with the procedures prior to participating in a walkthrough session.

Moderator and Scribe Responsibilities:

The moderator and scribe are responsible for the following activities prior to the walkthrough session.

Step	Activity
1	Review the materials provided by the presenter to become familiar with the contents.
2	Review the agenda and discuss any questions with the presenter.
3	Note directly on the review materials any nontechnical errors found during the review, such as spelling or typographical mistakes. While these errors are not discussed during the walkthrough, they should be provided to the author at the conclusion of the walkthrough.
4	Review the procedures (groundrules) for the structured walkthrough process. Clarify specific roles and responsibilities with the presenter. The moderator and scribe should be familiar with the procedures prior to participating in a walkthrough session.

Moderator's Responsibilities: The moderator is responsible for the following activities during the walkthrough session.

Step	Activity
1	Call the walkthrough session to order. It is important to start the session at the scheduled time.
2	Ask participants to introduce themselves and state their current responsibility/job assignment.
3	Briefly review the procedures and agenda for the walkthrough session.
4	Facilitate the walkthrough session. Every attempt should be made to adhere to the agenda and the established meeting procedures. Encourage active participation of all reviewers. Limit discussion to the identification of errors. The discussion of solutions is not part of the walkthrough process. Limit the author's participation to observation and answering questions. If the session exceeds 2 hours, the moderator should stop the session at a logical breaking point and schedule another session to continue the discussion. When walkthrough sessions exceed 2 hours, the productivity and attention span of the reviewers will be adversely affected.

Moderator's Responsibilities, continued:

Step	Activity
5	At the conclusion of the session, ask the reviewers to make a decision about the status of the work product as follows: A Accept product as is B Revise--no further walkthroughs for this product C Revise and schedule another walkthrough A majority opinion decides the action. If a majority opinion or consensus cannot be reached, the presenter will make the decision. If another walkthrough is necessary, the entire structured walkthrough process should be repeated.
6	Adjourn the walkthrough session at the scheduled time. If the agenda has not been completed, schedule a follow-up session.

Presenter's Responsibilities: The presenter is responsible for the following activities during the walkthrough session.

Step	Activity
1	Provide a brief overview of the work product.
2	If necessary, review outstanding issues from previous walkthrough(s).
3	Present the product to be reviewed. Answer reviewers' questions. The presenter can ask the author for assistance in answering questions.
4	At the conclusion of the meeting, if the reviewers cannot reach consensus about the status of the work product, the presenter is responsible for making that decision. The status will be one of the following: A Accept product as is B Revise--no further walkthroughs for this product C Revise and schedule another walkthrough

Scribe's Responsibilities: The scribe is responsible for the following activities during the walkthrough session.

Step	Activity
1	Record the beginning time for the walkthrough session.
2	Record the attendance of each participant.
3	Record the amount of time each reviewer spent reviewing the work product.
4	Record the technical errors identified by the reviewers. Record all significant comments and suggestions made by the reviewers and presenter.
5	Record suggested action items and other follow-up activities.
6	Record the end time for the walkthrough session.

Reviewers' Responsibilities: Each reviewer is responsible for the following activities during the walkthrough session.

Step	Activity
1	Provide the scribe with the time spent reviewing the work product.
2	Provide the appropriate introduction information (e.g., name and current job responsibilities).
3	Describe technical errors found during review of the work product. Be an active participant.
4	Ask questions as needed to clarify information about the work product.
5	Make constructive suggestions and comments about the work product.
6	Participate in the decision about the status of the work product: A Accept product as is B Revise--no further walkthroughs for this product C Revise and schedule another walkthrough If consensus cannot be reached by the reviewers, the presenter is responsible for making the decision.
7	Inform the author about any nontechnical errors found during the review by providing a marked up copy of the review package.

Scribe's Responsibilities: The scribe is responsible for the following activities after the walkthrough has taken place.

Step	Activity
1	Prepare the meeting record for the walkthrough session. Include any action items identified by the reviewers and the person/team responsible for completing each action item.
2	Circulate the meeting record to the participants for their review and comments.
3	Update the meeting record as needed. Distribute the revised meeting record to the author. Copies of the meeting record should be distributed to the other participants only if an additional walkthrough is required.

Reviewers' Responsibilities: The reviewers are responsible for the following activities after the walkthrough.

Step	Activity
1	Review the walkthrough session meeting record for accuracy and completeness.
2	Indicate changes that are needed to add or clarify information in the meeting record. Submit any changes to the scribe. If necessary, discuss discrepancies with the presenter.
3	If requested by the author of the work product, provide additional explanation of walkthrough comments.

Presenter's Responsibilities: The presenter is responsible for the following activities after the walkthrough.

Step	Activity
1	Review the walkthrough session meeting record for accuracy and completeness.
2	Indicate changes to the meeting record and return to scribe. If necessary, discuss discrepancies with the reviewers.
3	Initiate follow-up activities recommended by the reviewers. Verify that all action items have been assigned to the appropriate person/team.
4	Complete a Structured Walkthrough Management Summary Report. Include the following information: • Description of the work product reviewed. • Description of findings. In addition to findings, include significant problems that would cause schedule slippage or project cost increase. • Date, time, and duration of the walkthrough. • List of attendees. • Status decision (i.e.; accept as is, revise--no further walkthrough, or revise and schedule another walkthrough) and any other follow-up activities.
5	Distribute copies of the Structured Walkthrough Management Summary Report to the appropriate management personnel including the Project Manager and the Quality Assurance Team Manager.
6	Track progress made on open action items. As action items are closed, indicate closed status on the meeting record.
7	If necessary, schedule a follow-up walkthrough when the revised work product is ready.

Author's Responsibilities: The author is responsible for the following activities after the walkthrough.

Step	Activity
1	Make all necessary changes to the work product.
2	Use the structured walkthrough meeting record as a checklist to make sure all errors are corrected, reviewers comments have been addressed, and open issues are investigated.
3	Check with the presenter and reviewers, as needed, to obtain additional information or clarifications.
4	Conduct follow-up meetings with subject matter experts, as needed, to complete work product.
5	Prepare work product and participate in follow-up walkthrough, if required.

Quality Assurance Team Manager: The Quality Assurance Team Manager is responsible for the following activities after the walkthrough.

Step	Activity
1	Prepare a summary of the information contained in the Structured Walkthrough Management Summary Report.
2	Distribute the summary to the Technical Monitor for the software engineering task. The data presented in the report is included in monthly management reports.

Project Manager: The Project Manager is responsible for the following activities after the walkthrough.

Step	Activity
1	Review the Structured Walkthrough Management Summary Report.
2	If a problem exists that would cause a schedule slippage or project cost increase, send written notification to the project's Technical Monitor. An electronic mail message is an acceptable form of notification.
3	File the Structured Walkthrough Management Summary Report in the project management notebook/files.
4	Follow up on any action items that remain open. A formal plan may need to be developed for action items that cannot be resolved during the current lifecycle stage.

Project Technical Monitor:

The Project Technical Monitor is responsible for the following activities after the walkthrough.

Step	Activity
1	Review the summary information about the completed structured walkthroughs that is provided by the Quality Assurance Team Manager.
2	The Technical Monitor reports walkthrough statistics to appropriate levels of DOE senior management.

Preparation of Summary Report:

The presenter is responsible for the preparation of the Structured Walkthrough Management Summary Report (Summary Report). The presenter may ask the scribe to prepare the report. If the scribe prepares the report, the presenter reviews the report before it is distributed. A sample Summary Report is provided at the end of this appendix.

The Summary Report is distributed to the appropriate project personnel including:

- Project Manager
- Quality Assurance Team Manager

The Summary Report is used by the Quality Assurance Team to maintain statistical data on structured walkthroughs.

The Summary Reports generated during each stage of the software lifecycle will be checked during the In-Stage Assessments (ISAs). The purpose of the ISA check is to verify that structured walkthroughs were conducted during each lifecycle stage, that the walkthrough action items were documented, and that the action items have been properly resolved and closed.

Follow-up Walkthrough:

If a follow-up walkthrough is required, the procedures used in the original walkthrough should be repeated. Use the meeting record from the previous walkthrough as a checklist to confirm that the previously identified errors and issues were resolved.

Introduction: Structured walkthroughs are generally used to review software products or systems under development or maintenance at various lifecycle stages. This section describes the work products that should be reviewed at each stage of the lifecycle. The work products correspond to the deliverables described in the *Department of Energy Software Engineering Methodology*.

Planning Stage: The Planning Stage defines the work to be accomplished for a development or maintenance task and estimates the resources that will be required. During the Planning Stage, a structured walkthrough should be conducted for each project management tool.

Purpose	**Participants**
Reviews the project management tools for the project, such as the following: • Feasibility Study/Statement • Project Plan • Deliverables log • Milestone schedule • Work Breakdown Structure • Software Quality Assurance Plan	The developer and at least one systems analyst, preferably outside the project. If the project involves tele-communications (e.g., LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Requirements Definition Stage: The Requirements Definition Stage determines the scope and requirements for a development or maintenance project. During the Requirements Definition Stage, structured walkthroughs are used to identify problems, inaccuracies, ambiguities, and omissions in the Requirements Specifications.

Purpose	**Participants**
Reviews the following project work products: • Configuration Management Plan • Requirements Specification - Client goals - Data requirements - Functional requirements - Performance requirements - Interface requirements - Telecommunications - Computer security • Continuity of Operations Plan • Project Test Plan • Acceptance Test Plan	One or more of the project designers and at least one systems analyst. If the project involves telecommunications (e.g., LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Functional Design Stage:

The Functional Design Stage selects the design elements that determine how the software product will be constructed to meet the functional requirements. During the Functional Design Stage, the structured walkthroughs are used to identify flaws, weaknesses, errors, and omissions in the architecture of the design.

Purpose	**Participants**
Reviews the Functional Design Document, logical model, data dictionary, and requirements traceability matrix for errors in the following design areas: • Hardware • Software • Logical design • Telecommunications • System interfaces • Backup and recovery • Security • User interface • Reports	At least one systems analyst and one or more of the project designers/programmers. If the project involves tele-communications (e.g., LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

System Design Stage:

The System Design Stage uses the concepts and the system architecture to describe the system components in detail. During the System Design Stage, structured walkthroughs are used to review detailed specifications, and plans that address testing and implementation issues.

Purpose	**Participants**
Reviews the following work products: • Physical Model • Programming Standards • Program Specifications • System Design Document • Conversion Plan • Integration Test Plan • System Test Plan	At least one systems analyst and one or more of the project designers/programmers. If the project involves tele-communications (LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Programming Stage:

The Programming Stage involves the construction of the software product and the testing that is an integral part of the construction process. During the Programming Stage, walkthroughs are conducted on clean compiles, test data bases, and the operating documentation.

Purpose	**Participants**
Reviews listing (clean compiles) for approximately 80 hours of coding, or at the completion of a logical unit of work. Reviews should verify adherence to the following: • System design • Programming standards • Program Specifications • Software Configuration Management Plan	Technical personnel with appropriate expertise and at least two additional reviewers. The entire programming team might attend the walkthrough, depending on the approach. If the project involves tele-communications (e.g, LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.
Determines the adequacy of the Integration and System test data base and test plans.	

Software Integration and Testing Stage:

The Software Integration and Testing Stage is the transition from individual software components to an integrated software product. During the Software Integration and Testing Stage, structured walkthroughs are used to test the integrated product, check the accuracy of the operating documents that will be provided to the user(s) and maintenance programmer(s), and plan for the acceptance activities.

Purpose	**Participants**
Reviews the following documents: • Integration and System Test Results/Reports • Users Manual • Programmers Reference Manual • Installation Plan • Acceptance Test Plan • Training Plan • Preacceptance Checklists Reviews the following production activities: • Data conversion • Installation procedures	Participants include personnel with appropriate technical expertise and a technical writer. If the software product is an application on a mainframe platform, include a representative from Computer Operations. If the project involves tele-communications (e.g., LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.

Installation and Acceptance Stage:

The Installation and Acceptance Stage is the transition from a software product in development to a product or system in full production status. During the Installation and Acceptance Stage, structured walkthroughs are used to check the Acceptance Test Report and inspect the plans for activities performed in preparation for full-scale production.

Purpose	**Participants**
Reviews the following documents: • Acceptance Test Results/Report • Acceptance Checklist	Participants include personnel with appropriate technical expertise. If the software product is an application on a mainframe platform, include a representative from Computer Operations. If the project involves tele-communications (e.g., LAN or dial-up), include a representative from the appropriate functional area. If the project involves classified data or sensitive unclassified data, include a representative from the appropriate computer security program.
Performs the following production activities: • Data conversion • Installation procedures	

Types of Documents:

Structured walkthroughs are appropriate for reviewing other types of documents, such as the following:

- Departmental and contractual publications
- Long-range plans
- Administrative and technical operating procedures
- Technical reports
- Presentations

Types of Verification:

When reviewing other types of documents, structured walkthroughs are used to verify the technical and editorial accuracy and appropriateness of the content and format.

Purpose	**Participants**
Reviews for accuracy including the following: • Consistency • Completeness • Conformance to standards and guidelines • Style • Grammar and spelling	Technical experts, technical writer, and graphics expert.

Structured Walkthrough Meeting Record

Part 1: Structured Walkthrough Meeting Information

Complete the following blocks to record the structured walkthrough meeting information. If you have any questions, call your Quality Assurance Team representative.

Walkthrough Information	**Participant Information**		
	Participant Name/ Responsibilities	**Preparation Time**	**Present (✓)**
Project name:	Author:		
Walkthrough date:	Presenter:		
Walkthrough start time:	Moderator:		
Walkthrough end time:	Scribe:		
Walkthrough work product decision:	Reviewer:		
☐ Acceptable as presented	Reviewer:		
☐ Acceptable with revisions	Reviewer:		
☐ Revisions with another walkthrough	Reviewer:		
If a follow-up walkthrough is required:	Reviewer:		
Date: Location: Time:	Reviewer:		

Continued on next page

Structured Walkthrough Meeting Record

Part 2: Structured Walkthrough Findings

Record reviewer comments and action items identified during the structured walkthrough in the blocks below. If more space is needed, make additional copies of this page.

Number	Comments and Action Items	Date Closed

Structured Walkthrough
Management Summary Report

Product Name:

Product/Module Number: Acronym:

Work Product (check one)

___ Feasibility Study/Statement
___ Project Plan
___ Software Quality Assurance Plan
___ Software Configuration Management Plan
___ Continuity of Operations Plan/Statement
___ Data Dictionary
___ Requirements Traceability Matrix
___ Software Requirements Specification
___ Project Test Plan
___ Acceptance Test Plan
___ Design Methodology
___ Logical Model
___ Functional Design Document
___ Physical Model
___ Integration Test Plan
___ System Test Plan
___ Conversion Plan
___ System Design Document
___ Program Specifications
___ Programming Standards
___ Acquisition Plan
___ Installation Plan
___ Source Code
___ Transition Plan
___ Users Manual
___ Programmers Reference Manual
___ Training Plan
___ Other (specify):

Description of Product Reviewed:

Summary of Findings: (Include significant problems that would cause schedule slippage or project cost increase)

Date: Start Time: End Time: Duration:

Reviewers:

Presenter: _____ _____

Moderator: _____ _____

Scribe: _____ _____

Decision: A = Accept product as is
B = Revise--no further walkthrough for this product
C = Revise and schedule another walkthrough

Enter appropriate letter: _____

Software Engineering Methodology

Appendix D
In-Stage Assessment Process Guide

Table of Contents

Section **Page**

1.0 Overview D.1-1
Introduction D.1-1
Purpose D.1-1
Who Conducts D.1-1
Applicability D.1-1
Timing/Frequency D.1-1
Diagram D.1-2
In-Stage ISA D.1-2
End of Stage ISA D.1-2
Ownership D.1-3
Change Control D.1-4
Relationship to Other System Development Processes D.1-4
Process Measurements D.1-4

2.0 Process D.2-1
Scope D.2-1
Customers D.2-1
Suppliers D.2-1
Input D.2-1
Diagram D.2-2
Plan Review D.2-3
Schedule Review D.2-3
Receive Deliverables D.2-3
Conduct Review D.2-3
Review Levels D.2-3
Prepare Review Findings D.2-4
Process Output D.2-4
Responsibility Matrix D.2-7

3.0 Review Guide D.3-1
3.1 Reviewer Selection D.3-2
3.2 SEM Deliverables Review Guidance D.3-3
3.2.1 Project Plan D.3-7
3.2.2 Structured Walkthrough Meeting Record (documentation) (level 2) . . . D.3-7
3.2.3 Feasibility Statement (level 2) D.3-7
3.2.3a Software Quality Assurance Plan (level 3) D.3-8
3.2.4 Requirements Specification (level 3) D.3-8
3.2.5 Configuration Management Plan (level 2) D.3-8
3.2.6 Project Test Plan (level 3) D.3-9

Table of Contents (continued)

Section **Page**

3.2.7 Continuity of Operations (level 2) D.3-9
3.2.7a Acceptance Test Plan (draft) (level 2) D.3-10
3.2.8 Logical Model (level 2) D.3-10
3.2.9 Data Dictionary (level 2) D.3-10
3.2.10 Traceability Matrix (level 2) D.3-11
3.2.11 Functional Design Document (level 2) D.3-11
3.2.12 Program Specifications (level 2) D.3-12
3.2.13 Physical Model (level 2) D.3-12
3.2.14 Integration Test Plan (level 3) D.3-13
3.2.15 Conversion Plan (level 2) D.3-13
3.2.16 System Design Document (level 2) D.3-13
3.2.17 System Test Plan (level 3) D.3-14
3.2.18 Software Baseline (level 1) D.3-14
3.2.19 Acquisition Plan (level 2) D.3-14
3.2.20 Section Removed D.3-14
3.2.21 Transition Plan (level 2) D.3-15
3.2.22 User's Guide/Manual (level 2) D.3-15
3.2.23 Operating Documentation (level 2) D.3-16
3.2.24 Test Reports (level 2) D.3-16
3.2.25 Training Plan (level 2) D.3-17
3.2.26 Preacceptance Checklist (level 2) D.3-17
3.2.26a Installation Plan (level 2) D.3-17
3.2.27 Acceptance Test Plan (final) (level 2) D.3-18
3.2.28 Operational System (level 1) D.3-18
3.2.29 Acceptance Test Report (level 2) D.3-18
3.2.30 Maintenance Plan (level 2) D.3-19
3.3 Project Management Review Guidance D.3-20
3.3.1 Project Objectives Summary (level 1) D.3-22
3.3.2 Development Approach (level 3) D.3-22
3.3.3 Project Team (level 3) D.3-22
3.3.4 Roles/Responsibilities (level 2) D.3-23
3.3.5 Problem Escalation (level 1) D.3-23
3.3.6 Assumptions/Constraints/Dependencies (level 3) D.3-24
3.3.7 Estimates (level 3) D.3-24
3.3.8 Stage/Project Schedule (level 3) D.3-25
3.3.9 Status Reporting (level 3) D.3-25
3.3.10 Resource Planning (level 3) D.3-26
3.3.11 Budget - Plan Versus Actual (level 2) D.3-26
3.3.12 Sign-Offs (prior stage exit) (level 1) D.3-27
3.3.13 Configuration Management (level 2) D.3-27

Table of Contents (continued)

Section **Page**

3.3.13.1 SCI Identification (level 2) D.3-27
3.3.13.2 Change Initiation (level 3) D.3-28
3.3.13.3 Change Evaluation (level 3) D.3-28
3.3.13.4 Change Approval (level 3) D.3-28
3.3.13.5 Auditing (level 2) D.3-29
3.3.13.6 Change Control Log (level 2) D.3-29
3.3.13.7 Re-baseline Requirements (level 2) D.3-29

Revision date: 10/12/95
Author: Andy Bicocchi
Section(s): Change Control Page
Page number(s): D-iv
Summary of change(s): Added this change control page.

Revision date: 10/30/95
Author: Andy Bicocchi

a. Section(s): Throughout document
   Page number(s): Throughout document
   Summary of change(s): Revise the process to align it with the Software Engineering Methodology (SEM), which is replacing the System Development Guide (SDG).

b. Section(s): 1.0 Who Conducts
   Page number(s): D.1-1
   Summary of change(s): Added that the Quality Assurance Analyst will conduct ISAs.

c. Section(s): Entire document
   Page number(s): Most
   Summary of change(s): Revised document to ensure alignment to, and consistency with, the SEM and a Departmentwide focus (versus a Headquarters focus)

d. Section(s): 1.0 Overview, End-of-Stage ISA
   Page number(s): D.1-2
   Summary of change(s): Revised wording to reflect that copies of the report may be provided, as appropriate, to persons other than the project manager.

e. Section(s): 2.0 ISA Report
   Page number(s): D.2-6
   Summary of change(s): Added Corrective Action Recommendations section to the ISA report.

Revision date:
Author:

Section: **1.0**
Overview

Introduction: In-Stage Assessments (ISAs) are independent reviews of DOE system development and maintenance projects. ISAs are conducted in all stages of the software development and maintenance lifecycle (SDLC) in accordance with the ISA schedule, which is documented in the project plan. This document defines the process for planning and conducting ISAs.

Purpose: The purpose of ISAs is to assure, via an independent assessment, that the established system development and project management processes and procedures are being followed effectively, and that exposures and risks to the current project plan are identified and addressed.

An ISA is a project review that is conducted by a reviewer independent of the project. The reviewer assesses a project's processes, work products, and deliverables to verify adherence to standards and that sound system development and project management practices are being followed. An ISA is a paper review and does not require meetings among the involved parties.

Who Conducts: Within the current framework of deployment of this process, the Quality Assurance Analyst will conduct the ISAs.

Applicability: This process is applicable to all DOE development and maintenance efforts that would follow the DOE Software Engineering Methodology (SEM)[1].

Timing/ Frequency: An ISA can be conducted anytime during a stage whenever a deliverable is stable enough, or near the end of a stage to prepare for stage exit.

[1] The SEM is the standard for all DOE projects.

Diagram: The following diagram shows the timing of ISAs relative to the SEM lifecycle. The break out of the ISAs shown in the planning stage also would apply to the other SDLC stages.

```
    ISA
    |
    |   ISA
    |   |
    |   |Stage exit (this is a separate process)
    |   |▾
T   ▾   ▾T        T        T        T        T        T        T
*   T   T*   T   T*   T   T*   T   T*   T   T*   T   T*   T   T*
/)))2)))23)))2)))23)))2)))23)))2)))23)))2)))23)))2)))23)))2)))21
*Planning*Reqrmts *Func Dsg*Sys Dsgn*Programg*Soft Int*Installa*
R        RDefinit R        R        R        R& TestngR & AcptnR

S)))))))))))))))))))))) S D L C ))))))))))))))))))))))))))))))Q▸
```

In-Stage ISA: One or more in-stage ISAs can be scheduled for a given stage of development. The purpose of this review is to assess one or more deliverables when development (of that deliverable) is far enough along to allow for review, and early enough to allow for revisions prior to stage exit. The results of the review are contained in a report that is submitted directly to the project manager.

End of Stage ISA: An end-of-stage ISA must be conducted near the end of each stage of development. The purpose of this review is to assess the readiness of a project to proceed to the next stage by reviewing all the deliverables for the current stage. The results of this review are contained in a report that is submitted to the project manager. Copies of the report may be provided to the next-level manager and the client or system owner, as appropriate. In order to exit the current stage of development, the project manager must develop an acceptable action plan to address any open issues or qualifications.

Process Ownership: The following table identifies the In-Stage Assessment process sponsor and process owner. The responsibilities of the process sponsor include approving the initial process definition document and changes during process improvement. The responsibilities of the process owner include initial process definition, process implementation, and conducting ongoing process improvement. The process was originally developed with the support of a cross-functional process team. Some of the team members have since moved on to other positions, and have been deleted from the table. New team members are being recruited.

The team developed the In-Stage Assessment process under the direction of the DOE Chief Information Officer. After initial implementation, some members of the team have met periodically to review process improvement recommendations.

NAME	RESPONSIBILITY	BUSINESS FUNCTION	E-MAIL	PHONE
Wayne Jones	Process Sponsor	Technical Monitor	wayne.jones@ hq.doe.gov	(301) 903-4655
Andy Bicocchi	Process Owner	Quality Assurance	andrew.bicocchi@ hq.doe.gov	(301) 903-0616
Lew Bachman	Consultant	Quality Assur.	lew.bachman@ hq.doe.gov	(301) 903-0880
Beth Willis	Consultant	Software Development	elizabeth.willis@ hq.doe.gov	(301) 903-0878

Change Control: The ISA process is a component of the SEM. Changes to this process will be instituted using the same change mechanism that has been implemented to process changes to the SEM. All requests for change should be directed to the Manager of the site Quality Team.

Relationship to Other System Development Processes: The ISA process is a primary component of the DOE SEM. Together with other processes it serves to assure a consistent and predictable outcome in the resulting software products. The ISA process is complementary to other processes such as Stage Exits and Structured Walkthroughs.

Process Measurements: Process measurements are required in order to understand, in a quantifiable manner, the effectiveness of a process at work. If measurements indicate the process is not working as designed, a causal analysis should be conducted to identify the root cause of the breakdown, and changes should be implemented via the process improvement team. The ISA process is considered to be effective (working as designed) if:

- All issues that must be resolved in the current stage are identified.
- Unmet project objectives can be attributed to issues documented in an ISA.
- All issues without an acceptable action plan become qualifications to exit the current stage of development.
- All issues are properly documented.

Specific procedures for capturing the data for the above measurements will be defined during process improvement.

Section: **2.0**
Process

Scope: The In-Stage Assessment process begins with the scheduling of the review and ends with the delivery of the report produced after the review.

It is the responsibility of the project manager to develop and implement solutions for the issues and qualifications documented during the review. The project manager must develop an appropriate action plan for each issue.

Customers: The customers of the ISA process are those individuals or organizations that will use the output of the ISA process. The primary customers of the process are:

- Project manager
- Project manager's manager
- Quality Assurance (QA)

Secondary customers of the process are:

- System owner
- User POC
- Client representative

Suppliers: The project team produces work products and deliverables which become input to the ISA process.

Input: The following are the minimum inputs to the ISA process:

- Software development lifecycle deliverable(s)

- Project plan developed during the planning stage, which includes the work breakdown structure and timeline in addition to other components

- Updated project plan revised during all subsequent stages

- Structured walkthrough minutes/conference records

Diagram: The following diagram depicts the ISA process flow.

```
                +)))))))))))))))))),
Planning        *                  *  ISA target dates
Stage           *    PLAN FOR      *  defined in the project
                *   THE REVIEW     *  plan for each stage
                *                  *
                .)))))))))o))))))))-
                          ▼
                +)))))))))))))))))),
All             *                  *  Set date
Stages          *    SCHEDULE      *  Secure agreement
  T             *   THE REVIEW     *
  *             *                  *
  *             .)))))))))o))))))))-
  *                         ▼
  *             +)))))))))))))))))),
  *             *                  *  Stage deliverables
  *             *     RECEIVE      *  Updated project plan
  *             *   DELIVERABLES   *  Walkthrough minutes
  *             *                  *  Other applicable items
  *             .)))))))))o))))))))-
  *                         ▼
  *             +)))))))))))))))))),
  *             *                  *  Review deliverables
  *             *    CONDUCT       *  Review project plan
  *             *   THE REVIEW     *  Review other items
  *             *                  *  Formulate assessment
  *             .)))))))))o))))))))-
  *                         ▼
  *             +)))))))))))))))))),
  *             *                  *  Assessment of risk
  *             *     PREPARE      *  to achieve plan
  ▼             * REVIEW FINDINGS  *  List of concerns
                *                  *  Recommendations
                .)))))))))o))))))))-
                          ▼
                  To Project Manager
```

Plan ISA(s): In the planning stage, the target date for conducting the ISA(s) at each stage of development is documented in the project plan. In-stage ISAs can be conducted at any logical point in the stage. End-of-stage ISAs should be scheduled near the end of a stage (e.g., 2 or 3 weeks ahead of the stage exit milestone). In-stage ISAs should be scheduled for the next stage only, since reviews for all subsequent stages might be difficult to plan in advance.

Schedule Review: In each stage, as soon as practical, the actual assessment point should be established and agreed-to, and the activity scheduled so that the project manager is aware and there are no surprises.

Receive Deliverables: The reviewer should be provided with a copy of the deliverables to be reviewed, plus the current project plan if it is an end-of-stage ISA. The deliverables will vary according to the project's stage of development. The System Development Review Level table in section 3.2 provides detailed review guidance.

Conduct Review: The reviewer should examine each lifecycle deliverable. The depth of the examination will vary according to the deliverable and the project's stage of development. Guidance to assist the reviewer is provided in Section 3.0, Review Guide.

Review Levels: The following are the levels of review that can be performed on the SEM deliverables. For each deliverable, specific guidance is provided in Section 3.0, Review Guide.

Level	Explanation
1	Verify the existence of the work product or deliverable. Review to assure the work product or deliverable exists and is complete.
2	Verify minimum content exists. Review to ensure the minimum level of information has been provided. Verify the existence of content by checking sections/headings.
3	Verify content is rational. Review to make judgements as to the quality and validity of the deliverable.

Prepare Review Findings: The reviewer should document the results of the ISA and produce the report described in the Process Output section.

Process Output: A report must be produced when the ISA process is executed. Each ISA requires a report even if no issues were identified during the review. The report should be brief with the focus on providing a clear statement of the issues(s); solutions may be suggested, but are the project manager's responsibility. The report should include the following elements:

- A written assessment of the current project plan in terms of the following:

  - Risk to schedule and budget
  - Risk for next stage
  - Risk for remainder of project

  Risk categories:

  Low - Potential or existing problems must be addressed to avoid an impact to the current project plan. This would also apply if no issues were identified.

  Medium - Problems exist that have a high probability of impacting the current project plan or other dependencies.

  High - Serious problems exist (without an acceptable plan to resolve) that have a high probability of impacting user acceptance, the current project plan, or other dependencies.

- A list of issues/concerns if any were formed during the review. An issue is logged if there is a problem without a visible plan for resolution. Once a list of issues has been compiled, it should be reviewed with the project manager to see if any new or additional information might mitigate or eliminate any of them. Remaining issues must be addressed with an action plan from the project manager. Issues from an end-of-stage ISA might become "qualifications" to exiting the current stage of development. Refer to the stage exit process documentation for additional information.

Process Output continued:

The following are some examples of issues.

- No description of the estimating methodology used.
- No definition of a change control mechanism.
- Signoff (concurrence) from the prior stage is not visible.
- Concern about the appropriateness of the process used to arrive at technical decisions. In this example, the reviewer may recommend an additional in-depth review by technical experts as an action item.

If no issues were identified, the report only needs to contain the name of the project, date of the review, reviewer name, and a statement that no issues were identified.

- Additional reviewer comments, as appropriate. These include suggestions and recommendations that would benefit the project. The reviewer is encouraged to provide this feedback based on his/her experience. Reviewer comments are provided for the benefit of the project manager and should not be logged as issues requiring an action plan. In certain cases the reviewer may also recommend a more in-depth review by an individual highly skilled in a certain area, to help validate technical decisions and conclusions.

For in-stage ISAs, the written report is distributed to the following:

- Project manager

For end-of-stage ISAs, the written report is distributed to the following:

- Project manager, with copies to:
  - Project manager's manager
  - Quality Assurance manager

ISA REPORT

Project Name _____

Stage _____ Date _____

Reviewer _____ Phone _____

No.	Issues/Concerns:	Resolved
__	_____	_____
__	_____	_____
__	_____	_____
__	_____	_____
__	_____	_____

No.	Corrective Action Recommendations:
__	_____
__	_____
__	_____
__	_____
__	_____

Assessment of risk to schedule:

Low	Medium	High	
☐	☐	☐	For the next stage
☐	☐	☐	For the remainder of the project

Responsibility Matrix: The following matrix defines the responsibilities for the various organizations involved in the ISA process.

	Project Team	Quality Assurance	ISA Reviewer (1)
Prepare deliverables	P		
Schedule ISA	P	R	
Conduct ISA	S	P	
Compile list of issues	R	P	
Prepare assessment		P	
Ensure issue resolution	P		
Monitor process effectiveness		P	
Continuous process improvement	S	P	

Legend

P = Perform R = Review S = Support

(1) As the process is currently implemented, the ISA is conducted by Quality Assurance. However, it is possible for the ISA to be conducted by other parties, e.g. a peer project manager. Section 3.1, Reviewer Selection, provides a description of the skills required to conduct ISAs.

Section: **3.0**
Review Guide

Introduction: This section contains guidance on selecting an ISA reviewer, the appropriate level of review to be performed for each deliverable (or work product), and what to look for in each. While some deliverables are prepared once during the applicable stage of system development or maintenance, others are subsequently updated as the project progresses. These variances are highlighted in the review level tables.

Guidance: The following guidance is provided in this section.

3.1 Reviewer Selection
3.2 Deliverables Review Guidance
3.3 Project Management Review Guidance

Section: **3.1**
Reviewer Selection

The review must be conducted by a person who does not report to the same organization as the development or maintenance team. This allows for an independent view of problems and issues that might exist and serves as a cross-training tool. The experience and skills required include:

- Hands-on experience planning and managing technically complex software development projects.
- Working knowledge of the DOE Software Engineering Methodology (SEM).
- Ability to deal with people and communicate well.

Individuals who typically have the technical background, experience, and skills required include team managers, area managers, project managers, project leaders, task leaders, quality assurance representatives.

Section: **3.2**
SEM Deliverables Review Guidance

Description: This section provides guidance on what to look for when reviewing and assessing the various SDLC deliverables and components of the project plan. Work products (deliverables) are produced throughout the SDLC. They serve to document all project related data and form the basis of understanding between all parties involved in developing systems. These deliverables are required at various stages of the SDLC.

For a detailed description of the SDLC deliverables refer to the SEM.

Guidance: The review guidance for the following deliverables is provided in this section.

3.2.1 Project Plan
3.2.2 Structured Walkthrough Meeting Record
3.2.3 Feasibility Statement
3.2.3a Software Quality Assurance Plan
3.2.4 Requirements Specification
3.2.5 Configuration Management Plan
3.2.6 Project Test Plan
3.2.7 Continuity of Operations
3.2.7a Acceptance Test Plan (draft)
3.2.8 Logical Model
3.2.9 Data Dictionary
3.2.10 Cross Reference Matrix
3.2.11 Preliminary Design Document
3.2.12 Program Specifications
3.2.13 Physical Model
3.2.14 Integration Test Plan
3.2.15 Conversion Plan
3.2.16 System Design Document
3.2.17 System Test Plan
3.2.18 Software Baseline
3.2.19 Acquisition Plan
3.2.20 Section Removed
3.2.21 Transition Plan
3.2.22 User's Guide/Manual
3.2.23 Operating Documentation
3.2.24 Test Reports
3.2.25 Training Plan
3.2.26 Preacceptance Checklist
3.2.26a Installation Plan
3.2.27 Acceptance Test Plan (final)

3.2.28 Operational System
3.2.29 Acceptance Test Report
3.2.30 Maintenance Plan

The table on the following page identifies the appropriate level of review to be performed depending on the deliverable and the stage of development.

System Development Deliverables Review Level Table

TABLE LEGEND	SEM DELIVERABLE	SEM STAGE						
		Planning	Requirements Definition	Functional Design	System Design	Programming	Software Integration & Test	Installation & Acceptance
Review Levels: - 1 - Verify the existence of the deliverable - 2 - Verify minimum content exists - 3 - Verify content is rational (1) See Project Management Review Level table for details	Project Plan (1)	3	3	3	3	3	3	3
	Walkthrough Records	2	2	2	2	2	2	2
	Quality Assurance Plan	3						
	Feasibility Statement	2						
	Requirements Specification		3					
	Configuration Mgmt. Plan		2					
	Project Test Plan		3					
	Acceptance Test Plan (draft)		2					
	Continuity of Ops. Plan		2					
	Logical Model			2				
	Traceability Matrix			2				
	Data Dictionary			2				
	Functional Design Document			2				
	Program Specifications				2			
	Physical Model				2			
	System Design Doc.				2			
	Conversion Plan (as req.)				3			
	System Test Plan				3			
	Integration Test Plan				3			
	Software Baseline					1		
	Acquisition Plan					2		
	Transition Plan					2		

TABLE LEGEND	SEM DELIVERABLE	SEM STAGE						
		Planning	Requirements Definition	Functional Design	System Design	Programming	Software Integration & Test	Installation & Acceptance
	User's Guide					2		
	Integration Test Plan (final)					3		
	Installation Plan					3		
	System Test Plan (final)					3		
	Operating Documentation					2		
	Test Reports						2	
	Training Plan						2	
	Pre-Acceptance Checklist						2	
	Acceptance Test Plan (final)						3	
	Operating Documentat. (final)						2	
	Installation Plan (final)						3	
	Operational System							1
	Acceptance Test Report							1
	Maintenance Plan							2

Section: **3.2.1**
Project Plan

Refer to Section 3.3, Project Management Review Guidance, for project plan guidance.

Section: **3.2.2**
Structured Walkthrough Meeting Record (documentation) (level 2)

The structured walkthrough meeting record (i.e. the documented results of the walkthrough), including the Management Summary and any other associated available documentation, should contain a list of any defects identified during that particular walkthrough, and a clear disposition for each defect. The reviewer's objective is to verify the following:

- One or more structured walkthroughs were conducted depending on the stage of development in accordance with SEM guidelines.
- All defects have been addressed and closed; no action item has been left open-ended.
- If a followup structured walkthrough was required, the walkthrough was conducted and all action items were addressed.

Refer to Appendix C, Conducting Structured Walkthroughs, for additional information about structured walkthroughs.

Section: **3.2.3**
Feasibility Statement (level 2)

The feasibility statement is reviewed to determine if the following elements have been identified.

- Project objectives
- System automation alternatives
- Potential technical solutions
- Benefits and costs
- Recommendations

The project file should be reviewed to verify that a structured walkthrough was conducted and any action items resulting from the walkthrough were addressed or resolved.

Section: **3.2.3a**
Software Quality Assurance Plan (level 3)

The Software Quality Assurance Plan should clearly define the project's quality assurance policies and procedures. The plan should address the following:

- When In-Stage assessments will be conducted and by whom.
- Applicability of published standards and procedures.
- Monitoring for application of applicable standards and procedures.
- Assurance of resolution of discrepancies.
- Assessment of project progress.
- Assuring the integrity of the software product.

Section: **3.2.4**
Requirements Specification (level 3)

The requirements document is reviewed (use random sampling as appropriate) to assure that it includes the following elements.

- Project objectives are consistent with the objectives identified in the management plan.
- Requirements exhibit good attributes including clarity (no ambiguity), statement of a business problem or need (not the solution), and sufficient detail to allow for testing.
- All system requirements are addressed including data, functional, operational, security, telecommunications, and implementation requirements.

Section: **3.2.5**
Configuration Management Plan (level 2)

Refer to Section 3.3, Project Management Review Guidance, for configuration management plan guidance.

Section: **3.2.6**
Project Test Plan (level 3)

This document is reviewed to assure that it includes the following elements. (System Test plan details should be provided in the System Design Stage)

- Approach to testing (e.g. "...system testing will be conducted by an independent group...")
- Purpose and scope of test efforts to be conducted
- List of items that are planned to be tested, and items that will not be tested - Rationale for not testing what is not going to be tested
- Who (organization) that will be responsible for conducting the testing
- Which levels of testing are planned to be conducted (e.g. unit, integration, system, function, acceptance)
- Physical location(s) where testing is planned to be conducted
- List of known requirements for conducting testing activities (e.g. hardware, software, skills, space, equipment)
- Who is expected to sign-off and approve the tests

Section: **3.2.7**
Continuity of Operations Plan/Statement (level 2)

The continuity of operations plan (for mission essential applications) or statement (for non-mission essential applications) is reviewed to determine if it includes the following elements.

- Requirements for continuity of operation, such as data backup, data recovery, and operation startup
- Plans for backup and recovery operations
- Training requirements and plans so that the required skills will be in place to execute backup and recovery operations

Section: **3.2.7a**
Acceptance Test Plan (draft) (level 2)

The acceptance test plan should include the following elements.

- The timeframe when the test(s) are being planned to be conducted.
- A list of organization(s) from which individuals will conduct the tests. At a minimum, this should contain positions and skills required.
- The test cases and scenarios that are planned to demonstrate that the requirements have been satisfied in the new system or application.
- A list of, or reference to, requirements that the system should satisfy.
- Required hardware, software, documentation, special environmental or operational requirements, and any other special considerations (e.g.travel).

Section: **3.2.8**
Logical Model (level 2)

The logical model is reviewed to ensure it contains the following elements:

- Description of the final sources and destination of data
- Description of the net flow of data across the system boundary
- Complete picture of the system processes, data flow, and data stores
- Clear connections between the various pieces of the model

Section: **3.2.9**
Data Dictionary (level 2)

The data dictionary (data model) is reviewed to enure that data elements are documented in detail to include attributes, known constraints, input sources, output destinations, and known formats. The dictionary can include business rules processing statistics, and cross-referencing information.

The reviewer should assure him/herself that techniques have been employed (e.g. structured walkthroughs with appropriate persons) to ensure all data elements have been identified up to this point.

Section: **3.2.10**
Traceability Matrix (level 2)

Each requirement must be traceable to one or more design entities. The matrix is examined to ensure that it allows for the design entities to be traced back to the project requirements, and for verification that all requirements are satisfied by the design.

Section: **3.2.11**
Functional Design Document (level 2)

The Functional Design Document is reviewed to assure that it describes the functions of the system in user terminology. It should be written from the user/system owner's perspective. It should enable the owner/users to understand how the design will satisfy the requirements, providing an opportunity to give feedback before the design is completed.

Under separate covers, or as sections of the Preliminary Design Document, the reviewer should ensure that the following design related work products have also been documented:

- The design alternatives including evaluation criteria, alternative descriptions, and recommendations. (Note: A reference to the alternatives study is acceptable).
- The operating environment including hardware, software, telecommunications, and interfaces.
- System design demonstrating the system architecture, system inputs, outputs, interfaces, and end-user interfaces.
- Design method used, design entities, and design dependencies.
- Security and control measures that will be incorporated into the software system.
- The display conventions that will be followed for the design of all end-user interfaces (such as application screens).
- The naming and identification conventions for programs, libraries, data bases, tables, and files.

- The implementation approach, reflecting the necessary planning to take the system through to implementation.

Section: **3.2.12**
Program Specifications (level 2)

This deliverable may consist of one or more documents. The specifications are reviewed to assure inclusion of the following elements.

- Program design characteristics including program architecture, and software and common program features.
- Standards and guidelines that will govern each program coded for the system.
- Program specifications (use sampling as appropriate - good rule of thumb is 15 percent). This should include a description of the logic and other particulars for each program in the system.

Section: **3.2.13**
Physical Model (level 2)

The physical model is reviewed to determine if it includes the following attributes and components:

- Contents and organization of the physical data files
- Processes and data flows among the files
- Modules or groups of modules structured into a hierarchy of sub-systems
- Manual tasks with physical input or output characteristics.

Section: **3.2.14**
Integration Test Plan (level 3)

This document (or section) is reviewed to assure that an incremental integration and test plan exists. It should include the following elements.

- Purpose and scope of test efforts to be conducted
- List of items to be tested, and items that will not be tested
  - Rationale for not testing what is not going to be tested
- Responsible individual (or organization) for each activity including sign-offs, management, and acceptance
- Schedules
- Description of "how" the testing will be conducted

Section: **3.2.15**
Conversion Plan (level 2)

The conversion plan addresses the process of transitioning from the existing manual or automated process to the new application or system. This plan is reviewed to assure that it includes the following elements:

- General information. This section should describe, in a concise manner, relevant information about the system and task.
- Data conversion. This section should describe the strategy and specific activities required to convert data from the old system to the new one.

Section: **3.2.16**
System Design Document (level 2)

The System Design Document is reviewed to ascertain whether it translates requirements into precise specifications of the software components, interfaces, and data which are necessary before coding and testing can begin. The sequence and conditions inherent within modules should be documented.

The requirements cross-reference matrix should be updated and should allow for each requirement to be traced to one or more of the detailed design entities to verify that all of the requirements will be satisfied by the detailed design.

Section: **3.2.17**
System Test Plan (level 3)

This document is reviewed to assure that it includes the following elements.

- Scope of the testing effort, and testing schedules.
- Objectives and definition of the test cases, and the hardware and software configuration for each test or set of tests.
- A requirements verification matrix mapping individual tests to specific requirements and specifying how each system requirement will be validated.
- Identification of test tools and test support needs (e.g. hardware / software to simulate production environment and conditions).
- Physical location(s) where testing is planned to be conducted.
- Who is expected to sign-off and approve the tests.
- How will fixes to defects be handled and how will re-testing be conducted.

Section: **3.2.18**
Software Baseline (level 1)

The reviewer should look for evidence that the software baseline (the system or application) has been completed. This may include a demo of the working application, a library listing showing all the coded programs, evidence of positive test results, and confirmation from the individuals who completed the testing activities.

Section: **3.2.19**
Acquisition Plan (level 2)

An acquisition plan should exist, if hardware, software, or services will need to be acquired at some point during the lifecycle in order to implement the system.

Section: **3.2.20**
Section Removed

Section: **3.2.21**
Transition Plan (level 2)

The Transition Plan should describe the detailed plans, procedures, and schedules that will guide the transition process to full operation of the system or application. It should demonstrate that it has been coordinated with operational and maintenance personnel. Refer to the DOE SEM document, Section 7.7, Plan Transition to Operational Status for more details.

Section: **3.2.22**
User's Guide / User Manual (level 2)

Note: The User's Guide may be produced as part of the Operating Documentation. The user's guide (or user manual) is reviewed to determine if it contains sufficient information and detailed instructions required to access and use the system functions. For very small systems, a quick reference card may be more appropriate and sufficient. For larger systems, a reference card may also be provided in addition to the users's guide. The extent of the user's guide may also depend on the depth of the online help provided; the lower the level of online help the less extensive the user's guide needs to be. A user's guide should include:

- Overview of the system history, background, architecture, and current version
- Complete coverage of all system functions, in a logical order
- Samples of screens and reports, where appropriate to show examples
- Instructions for installing, configuring, and accessing the system
- Security features including what is accessible to each category of users
- Who to contact for additional information or help

Additional note to reviewer: Has it been tested by the documentation team, and have their comments been addressed?

Section: **3.2.23**
Operating Documentation (level 2)

The system documentation should be complete, to allow for both ease of operation and maintenance. The following documentation should be produced.

- Programmer's Reference Manual. This document should include the information required to allow the programming staff to understand and maintain the system programs, data bases, interfaces, and operating environments.

- System Administration Manual. This document should provide information necessary to enable the system administrator to understand and access the system functions required to manage the system.

- Data Base Administration Manual. This manual should provide the information necessary to enable the data base administrator (DBA) to understand the logical and physical organization, and the record structure of the system.

- Operations Manual. This manual should provide the operations group with a description of the system operation environment and the detailed instructions they need to execute the system functions.

Section: **3.2.24**
Test Reports (level 2)

The project files should be reviewed to ensure that system and acceptance testing results have been documented. Any defects found should have been corrected according to the established procedures that should include the process for assigning, handling, and dispositioning defects.

Section: **3.2.25**
Training Plan (level 2)

This document is reviewed to determine if it includes the following elements in the level of detail needed for training staff to begin logistical training arrangements. A "draft" level document is not intended to provide complete training plan information.

- Background - system description, objectives, curriculum overview
- Training requirements - environment, audience, category, skill level of users
- Objective - expected results of the training in terms of the increased level of user knowledge
- Training strategy - type of training, schedule, duration, sites
- Training resources - resources required, responsibilities of involved parties
- Environment - facilities, support from other groups, equipment, actions required
- Training materials - types of materials required, e.g., system reference manual

Section: **3.2.26**
Preacceptance Checklist (level 2)

Refer to the DOE SEM document, Section 8.3, Initiate Acceptance Process, for details and a sample of the checklist.

Section: **3.2.26a**
Installation Plan (level 2)

This plan is reviewed to assure that it includes the following elements. See SEM section 7.2, Installation Plan, for additional details.

- General information relevant to the installation process.
- Assumptions and dependencies related to the installation activities.
- Strategy and schedule for phasing in the new system and disposing of the old one.

Section: **3.2.27**
Acceptance Test Plan (final) (level 2)

The acceptance test plan should include the following elements.

- The schedule for the test(s) that will be conducted.
- A list of the individuals who will conduct the tests and their position/organization.
- The test cases and scenarios designed to demonstrate that the requirements have been satisfied in the new system or application.
- A list of, or reference to, requirements that the system should satisfy.
- Required hardware, software, documentation, special environmental or operational requirements, and any other special considerations (e.g.travel).

Section: **3.2.28**
Operational System (level 1)

The reviewer needs to assure that all the system turnover activities have been completed, including the following items.

- Execution and completion of the DOE Acceptance process.
- If deemed appropriate, interviews with the client to determine satisfaction with requirements met.
- Final acceptance turnover meeting. This is often waived or cancelled.
- Acceptance checklist(s) completed as per the Acceptance process.

Section: **3.2.29**
Acceptance Test Report (level 2)

The formal Acceptance Test Report should include a summary of the test procedures executed, any problems detected and corrected, and the projected schedule for correcting any problem reports.

Section: **3.2.30**
Maintenance Plan (level 2)

Note: This may be part of the Transition Plan.
A plan for the support and maintenance of the system after turnover to the user(s) or after installation into its intended production environment should be documented. It should include the names of the person(s) and/or organization(s) that will provide support, a service level agreement, which process will be followed (presumably the DOE SEM) for software maintenance, and any schedules, if appropriate.

Section: **3.3**
Project Management Review Guidance

Description: Project management includes the set of activities related to planning, organizing, directing, staffing, and controlling available resources to achieve distinct goals and objectives established at the beginning of a project. The key to successful project management is a good project plan, which is developed initially in the planning stage. The plan is dynamic, and should be reviewed and revised in subsequent stages to reflect approved changes.

Guidance: This section is a reference guide to assessing the various components of a project plan. The guidance for the following deliverables is provided in this section.

3.3.1 Project objectives summary
3.3.2 Development approach
3.3.3 Project team
3.3.4 Roles/responsibilities
3.3.5 Problem escalation
3.3.6 Assumptions/constraints/dependencies
3.3.7 Estimates
3.3.8 Stage/project schedule
3.3.9 Status reporting
3.3.10 Resource planning
3.3.11 Budget - plan vs. actual
3.3.12 Sign-offs (prior stage exit)
3.3.13 Configuration management

The table on the following page identifies the appropriate level of review to be performed depending on the project management activity and the stage of development.

Project Management Review Level Table

TABLE LEGEND	PROJECT PLAN	SEM STAGE						
		Planning	Requirements Definition	Functional Design	System Design	Programming	Software Integration & Test	Installation & Acceptance
Review Levels - 1 - Verify the existence of the deliverable - 2 - Verify minimum content exists - 3 - Verify content is rational	Project objectives summary	1						
	Development approach	3						
	Project team	3			3	3	3	
	Roles/responsibilities	2			2	2	2	
	Problem escalation	1						
	Assumptions/constraints/ dependencies	3	3	3	3	3	3	
	Estimates	3	3	3	3	3	3	
	Stage/project schedule	3	3	3	3	3	3	
	Status reporting	3	1	1	1	1	1	
	Resource planning	3	1	1	1	1	1	
	Budget - plan vs. actual	2	2	2	2	2	2	
	Sign-offs (prior stage exit)		1	1	1	1	1	
	CONFIGURATION MGMT.							
	SCI Identification		2					
	Change initiation		3					
	Change evaluation		3					
	Change approval		3					
	Auditing			2	2	2	2	2
	Change control log			3	3	3		
	Re-baseline requirements			2	2	2		

Section: **3.3.1**
Project Objectives Summary (level 1)

This section defines the project objectives, in summary form. It is intended to give the reader a high level overview of the project. It should be in summary form, and therefore brief (e.g., one-half page to 2-3 pages). It can be copied from other project documents where it may already exist.

Section: **3.3.2**
Development Approach (level 3)

This section defines the development environment in terms of the SDLC methodology for the project. This section should include the following elements.

- The methodology that is followed for this project. The standard for DOE projects is the SEM.
- The variations that are taken (if any) from the standard development model.
- If deviations are taken, what is in place to mitigate the increased risk.
- Any feature that is unique regarding the SDLC or project management aspects of this project.

Section: **3.3.3**
Project Team (level 3)

The project team section of the project plan is reviewed to assure that the members of a project team have been identified, by name, for the next stage of development and, if possible, for the entire project. The composition of the team should be reviewed and revised as the project progresses through the various stages of development. This section should be reviewed in concert with the section on schedules and estimates to determine if the two sections support each other.

Section: **3.3.4**
Roles/Responsibilities (level 2)

The roles and responsibilities of all team members and the users should be documented initially in the Planning stage. The information should be revised in subsequent lifecycle stages if there are changes, and should be based on the skills and experience of each individual.

This section should include, at a minimum, the names and organizations of those individuals responsible for providing a concur/nonconcur position (sign-off) at stage exit. Typically, this would include the project manager's manager, the system owner, the quality assurance representative, and the user point of contact. Other persons having a support or participatory role should also be identified if possible.

Section: **3.3.5**
Problem Escalation (level 1)

All problems should be brought to the attention of the project manager first. Problem management is the instrument used to handle project related conflicts, misunderstandings, and problems that cannot be resolved at the project manager level. This section is reviewed to ensure the following items have been identified.

- Escalation point (individual(s) to whom the problem will be brought for resolution if the problem is not resolved at the project manager level.

- Timeframe for a problem to be brought to resolution after it was escalated. This could be one day, several days, or week(s).

Section: **3.3.6**
Assumptions/Constraints/Dependencies (level 3)

This section should contain all project-wide assumptions, known constraints, and dependencies identified for this project. Assumptions and dependencies should be very specific. Generally known or accepted practices usually do not need to be documented. In many cases, assumptions, constraints, and dependencies will affect project estimates and schedules. This section should be updated and reviewed at every stage of the SDLC.

The following is an example of a dependency:

> The development team is dependent on the availability of user personnel to review deliverables within ten (10) business days.

The following statement is an example of an assumption.

> One senior systems analyst with 4 years of Paradox experience will be on board as planned by August 1, 1994.

Section: **3.3.7**
Estimates (level 3)

Estimates are reviewed to assure they include the following elements. In subsequent stages, changes to the original estimates may be documented.

- Description of the methodology or combination of methodologies used to arrive at the estimates (planning stage only).
- Line items for all associated project costs including labor months and other direct costs.
- Breakout of project stage activities. This should be detailed for the next stage of development and in summary form for the rest of the project.
- Revisions from the prior stage of development, if there are approved changes that increase or decrease project cost.
- Assumptions upon which the estimates are based.
- Factors used to arrive at the contingency numbers.

Section: **3.3.8**
Stage/Project Schedule (level 3)

The project schedule for the next stage is reviewed for evidence of the following attributes. Each activity and its associated data (e.g., start and end dates) can be part of a work breakdown structure that also shows relationships.

- Date for the stage exit.
- Description of each deliverable or activity.
- Individual (name) responsible for the deliverable or activity.
- Projected start and end dates for each activity.
- Impact (if any) that approved changes have on the schedule of record (the one approved at the prior stage exit) and the baselining of the new schedule.
- Critical path(s) identified and dependencies, if any.

For subsequent stages, at a minimum, milestones (stage exit dates) and SEM deliverables should be documented for each stage.

Section: **3.3.9**
Status Reporting (level 3)

Regular, clear, and effective communication is critical to the success of any project. In the planning stage, this section is reviewed to assure the following elements have been identified.

- Method for reporting project progress and problems
- Frequency of status meetings
- Procedure for tracking actions items to closure

In subsequent stages, look for evidence that the established reporting practice is being followed. This might include conference records and meeting minutes.

Section: **3.3.10**
Resource Planning (level 3)

Resource planning is essential to the success of a project. It should be started as early as possible in the planning stage, and then revised for each subsequent stage of development. It should be based on the known project requirements and should demonstrate when necessary resources and skills need to be on board.

The following are examples of resources:

Hardware: 8 IBM PS/2s with 10 Mg of RAM and VGA monitors
Software: PS/2s must have IBM OS/2 2.1 with communications manager
Office space: four offices, 12 by 12 feet (each office will accommodate two programmers
People: two senior analysts, four programmers, two junior programmers, one quality assurance analyst (specify experience requirements)
Dates: target dates for acquiring specified resources

In subsequent stages, look for evidence that action has been taken to ensure resources will be available when needed.

Section: **3.3.11**
Budget - Plan Versus Actual (level 2)

Review this section to verify that it includes the following elements.

- Project status from the financial perspective.
- Planned/approved expenditure level to date.
- Actual expenditure level to date.
- The difference (delta) between planned and actual expenditure levels, if any, and to what this difference is attributable.

Any deltas should be appropriately reflected in the project plan.

Section: **3.3.12**
Sign-Offs (prior stage exit) (level 1)

The project files should contain the concurrences (sign-offs) received at the prior stage exit. These should be from the individuals identified as having concurrence rights in the roles and responsibilities section of the project plan.

Section: **3.3.13**
Configuration Management (level 2)

The configuration management section of the project plan has several important sections that need to be reviewed including the following items.

3.3.13.1 SCI identification
3.3.13.2 Change initiation
3.3.13.3 Change evaluation
3.3.13.4 Change approval
3.3.13.5 Auditing
3.3.13.6 Change control log
3.3.13.7 Re-baseline requirements

Section: **3.3.13.1**
SCI Identification (level 2)

Review this section to verify that software configuration item (SCI) baselines have been identified in order to establish control points for various SDLC deliverables. The following elements are required.

- Functional baseline (requirements specification)
- Design baseline (system/subsystem specifications)
- Production baseline (first version of code)
- Operations baseline (final version of code)

The training baseline is optional.

Section: **3.3.13.2**
Change Initiation (level 3)

Review this section to determine if it contains the following elements.

- The name and organization of the individual(s) who are authorized to approve requests for change. Ideally, this should be one person such as the user point-of-contact for the project.

- The form(s) on which the change request must be submitted. A specific form is not required. Quality Assurance can provide a sample of an appropriate form to use.

Section: **3.3.13.3**
Change Evaluation (level 3)

Review this section to assure that the following elements are identified.

- The name and organization of the individual(s) who is responsible to evaluate the request for change. In many cases, a change control board is responsible to determine whether a change can be contained within the current project plan.

- Change evaluation criteria such as the nature of the requested change, time estimates to perform, and an impact analysis on the current schedule and plan.

Section: **3.3.13.4**
Change Approval (level 3)

This section should define the procedure for change approval including the following items.

- The name and organization of the individual(s) who is authorized to make decisions as to the disposition of evaluated changes.

- The acceptable disposition of the change request; e.g., make the requested change, hold the change for a future release, or reject the change.

Section: **3.3.13.5**
Auditing (level 2)

This section deals with assuring that the SCM plan is being followed. Look for evidence that demonstrates that change against the previously identified baselines is being managed. Typically, all changes should be recorded in the project's change control log. Other complementary documents such as two-way memos, E-mail notes and messages, and internal forms might also exist.

Section: **3.3.13.6**
Change Control Log (level 2)

Review the project's change control log to:

- Confirm that change activity is being properly recorded and controlled
- Assure that all changes to the project baseline have been evaluated, approved, and noted.
- Assure that the impact of accepted changes has been adequately reflected in the project plan, particularly in revised estimates.

Section: **3.3.13.7**
Re-baseline Requirements (level 2)

Review this section to ensure that any changes, additions, or deletion of requirements are properly identified and recorded. The new baseline consists of the agreed-to set of requirements from the prior stage, plus or minus any requirements from the current stage of development. The project plan in terms of schedule and budget should reflect the new baseline.

Software Engineering Methodology

Appendix E
Stage Exit Process Guide

Table of Contents

Section		Page
1.0	Overview	E.1-1
	Introduction	E.1-1
	Purpose	E.1-1
	Applicability	E.1-1
	Timing/Frequency	E.1-1
	Diagram	E.1-2
	Process Ownership	E.1-2
	Change Control	E.1-3
	Relationship to Other System Development Processes	E.1-3
	Process Measurements	E.1-3
2.0	Process	E.2-1
	Scope	E.2-1
	Customers	E.2-1
	Suppliers	E.2-1
	Input	E.2-1
	Diagram	E.2-2
	Plan Stage Exits	E.2-3
	Schedule Exit	E.2-3
	Distribute Materials	E.2-3
	Receive Positions	E.2-4
	Prepare Action Plans	E.2-4
	Conduct Exit Meeting	E.2-4
	Output	E.2-5
	Meeting Outcome	E.2-5
	Management Briefs	E.2-5
	Responsibility Matrix	E.2-6

List of Examples

Example **Page**

1 Stage Exit Notification Memo E.2-8

2 Stage Exit Distribution List E.2-9

3 Stage Exit Participation - LAN Project E.2-10

4 Stage Exit Participation - Mainframe Project E.2-11

4A Stage Exit Participation - Client/Server Project E.2-12

5 Stage Exit Position Response Form - Sent to Approver E.2-13

6 Stage Exit Position Response Form - Concur With Qualifications E.2-14

7 Stage Exit Position Response Form - Non-Concur E.2-15

List of Templates

Stage Exit Notification Memo

Stage Exit Position Response Form

Revision date: 10/12/95
Author: Andy Bicocchi

a. Section(s): Throughout document
Page number(s): Throughout document
Summary of change(s): Revise the process to align it with the Software Engineering Methodology (SEM), which is replacing the System Development Guide (SDG).

b. Section(s): Throughout document
Page number(s): Throughout document
Summary of change(s): Changed Service Areas to the more generic Support Areas.

c. Section(s): Change Control Page
Page number(s): E-v
Summary of change(s): Added the change control page.

d. Section(s): Process measurements
Page number(s): E.1-3
Summary of change(s): Revised the process measurements section; added qualifications measurements.

e. Section(s): Responsibility Matrix
Page number(s): E.2-6
Summary of change(s): Updated the responsibility matrix to reflect current environment.

f. Section(s): Management Briefings
Page number(s): E.2-5
Summary of change(s): Added management briefings section - a new process activity.

g. Section(s): Diagram/Process Flow
Page number(s): E.2-2
Summary of change(s): Added management briefing box to the process flow diagram.

Revision date:
Author:

Section: **1.0**
Overview

Introduction: The DOE Software Engineering Methodology (SEM) describes the standard system development lifecycle (SDLC) methodology used for systems developed and maintained for the Department Of Energy. For better manageability and control, each system development effort is organized into logical, related segments called stages. Each stage must be officially exited (approved) before the next stage can begin. The decision points at the end of each stage are called Stage Exits.

A Stage Exit is the vehicle for securing the concurrence (i.e., approval) of designated individuals to continue with the project and move forward into the next stage of development or maintenance. The concurrence is an approval (sign-off) of the deliverables for the current stage of development including the project plan. It indicates that all qualifications (issues and concerns) have been closed or have an acceptable plan for resolution.

Purpose: The purpose of a Stage Exit is to:

- Allow all functional areas involved with the project to review the current project plan. This includes, at a minimum, a detailed plan for the next stage, and high-level plans for the remainder of the project.
- Provide a forum to raise qualifications (issues and concerns) if issues exist that will impact the project plan.
- Ensure an acceptable action plan exists for all qualifications raised.
- Obtain concurrence on current stage deliverables, and to begin the next stage of development.

Applicability: This process is applicable to all system development and maintenance efforts that follow the SEM.

Timing/ Frequency: A Stage Exit is conducted at the end of each stage of development or maintenance.

Diagram: The following figure shows the timing of Stage Exits, relative to the SDLC.

```
                        Stage Exits

              +)))))))))0)))))))))0)))))))))0)))))))))0)))))))))0)))))))))),
              *         *         *         *         *         *         *
              ▾         ▾         ▾         ▾         ▾         ▾         ▾

*             *         *         *         *         *         *         *
/)))))))))3)))))))))3)))))))))3)))))))))3)))))))))3)))))))))3)))))))))1
*Planning*Reqrmts *Fnc Dsgn*Sys Dsgn*Programg*Soft Int*Installa*
R         RDefinitnR         R         R         R& TestngR & AcptnR

S)))))))))))))))))))))))))))) S D L C )))))))))))))))))))))))))))Q▸
```

Process Ownership: The following table identifies the Stage Exit process sponsor and process owner. The responsibilities of the process sponsor include approving the initial process definition document and changes during process improvement. The responsibilities of the process owner include assuring the process is working once implemented, and conducting ongoing process improvement. The process was originally developed with the support of a cross-functional process team. Some of the team members have since moved on to other positions, and have been deleted from the table.

The team developed the Stage Exit process under the direction of the DOE Chief Information Officer. After initial implementation, some members of the team have met periodically to review process improvement recommendations.

NAME	RESPONSIBILITY	BUSINESS FUNCTION	E-MAIL	PHONE
Wayne Jones	Process Sponsor	Technical Monitor	wayne.jones@ hq.doe.gov	(301) 903-4655
Andy Bicocchi	Process Owner	Quality Assurance	andrew.bicocchi @hq.doe.gov	(301) 903-0616
Lew Bachman	Consultant	Quality Assur.	lew.bachman@ hq.doe.gov	(301) 903-0880
Beth Willis	Consultant	Software Development	elizabeth.willis@ hq.doe.gov	(301) 903-0878

Change Control: The Stage Exit process is a component of the SEM. Changes to this process will be instituted using the same change mechanism that has been implemented to process changes to the SEM. All requests for change should be directed to the Manager of the site Quality Team.

Relationship to Other System Development Processes: The Stage Exit process is a primary component of the DOE SDLC. Together with other processes it serves to assure a consistent and predictable outcome in the resulting software products. The Stage Exit process is complementary to other processes such as In-Stage Assessments and Structured Walkthroughs.

Process Measurements:

Defines the measurements that will allow for determining the effectiveness of the process at work. The main process measurement vehicle will be qualifications (issues) that are raised and closed. For each stage, the following qualification data will be collected:

- Quantity
- Severity Level
  - Low
  - Medium
  - High
- Qualifications Closed
- Qualifications Remaining Open

There are a number of tools that can be used to help track qualifications. These range from a word processor like Word Perfect to a project manager like Microsoft Project.

Section: **2.0**
Process

Scope: The Stage Exit process begins with a notification to the extended development team (e.g., system Owner, user point-of-contact, support areas) that a stage exit has been scheduled. The process ends with the receipt of concurrence from the designated approvers to proceed to the next stage. Concurrence indicates that all known issues have an acceptable plan for resolution.

Customers: The customers of the Stage Exit process are those individuals or organizations that will use the output of the process. The primary customers are:

- Software development department
- System owner
- User point of contact (POC)
- Quality Assurance (QA)

Suppliers: The following individuals or organizations provide input to the Stage Exit process:

- System owner
- Project manager's manager
- User POC
- QA
- Support areas

Input: The following are the minimum inputs to the Stage Exit process:

- System development lifecycle deliverable(s)
- Initial project plan developed during the planning stage
- Updated project plan revised during all subsequent stages
- Issues to be addressed (generic)
- Qualifications from the approvers
- Issues that remain open from the In-Stage Assessment

Diagram: The following diagram depicts the Stage Exit process flow.

```
                +)))))))))))))))))),
PLANNING        *                  *  Stage Exit milestones
STAGE           * PLAN PROJECT's   *  defined in the project
                *  STAGE EXITS     *  plan for each SEM stage
                *                  *
                .)))))))))))))))))-

                +)))))))))))))))))),
 ALL            *                  *  Schedule exit for the
STAGES          *    SCHEDULE      *  current stage; notify
                *  CURRENT EXIT    *  participants (memo)
   *            *                  *
   *            .))))))))0))))))))-
   *                     ▾
   *            +)))))))))))))))))),
   *            *                  *  Distribute current project
   *            *   DISTRIBUTE     *  plan and any other items
   *            *    CURRENT       *  needed to exit stage
   *            *  PROJECT PLAN    *
   *            .))))))))0))))))))-
   *                     ▾
   *            +)))))))))))))))))),
   *            *                  *  Receive positions and
   *            *     RECEIVE      *  qualifications (if any)
   *            *    POSITIONS     *  from participants
   *            *                  *
   *            .))))))))0))))))))-
   *                     ▾
   *            +)))))))))))))))))),
   *            *                  *  Prepare plans to resolve
   *            *     PREPARE      *  qualifications (if any
   *            *  ACTION PLANS    *  were raised)
   *            *                  *
   *            .))))))))0))))))))-
   *                     ▾
   *            +)))))))))))))))))),
   *            *                  *  Present plans for addressing
   *            *     CONDUCT      *  qualifications and secure
   ▾            * CLOSURE MEETING  *  concurrence
                *                  *
                .)))))))))))))))))-

                +)))))))))))))))))),
                *                  *  Provide a briefing of status, issues,
PERIODIC        *  BRIEF UPPER     *  risk, from the QA analyst's
                * LEVEL MANAGER    *  perspective
                *                  *
                .)))))))))))))))))-
```

Plan Stage Exits: In the Planning stage, the planned date for exiting each stage of development is identified and documented in the project plan. It is common practice for the Stage Exit date for the next stage to be more specific and the dates for subsequent stages to be high level milestones.

Schedule Exit: For each stage, as soon as practical, the actual Stage Exit date should be established and the exit meeting scheduled. Two or three weeks prior to the exit meeting, a memo is sent to all persons participating in the stage exit to communicate the following information.

- Notify participants that a stage exit has been scheduled. Participants include approvers (e.g., system owner), support area representatives (e.g., Network Engineering), and individuals with a need to know (e.g., contractor management).

- Request that the approvers provide feedback one week before the exit meeting. This will allow the project manager[1] time to work issues and develop action plans prior to the exit meeting.

- Invite participants to attend the exit meeting.

Examples of a memo, distribution list, and response forms are provided in the example section of this guide.

Distribute Materials: The current project plan, and any other material relevant to exiting the stage, should be distributed to the participants[2] along with the memo. Relevant materials include for example known issues, and unplanned deliverables.

The participants should be familiar with planned deliverables (e.g. the Requirements document in the Requirements Definition stage) since it is common practice for them to review drafts as they are developed. If this is not the case, then planned deliverables also need to be distributed at this time.

The project plan is dynamic and typically undergoing changes up to the last minute, and is distributed (together or under separate cover) at the same time as the stage exit notification memo.

---

[1] Project manager is the generic term for the person responsible for planning and day-to-day control of the project; e.g., task leader, team manager, or project leader.

[2] In the Planning Stage, participants include all support areas.

Receive Positions: A position is required from the list of approvers. This position can be concur, concur with qualifications, or non-concur. The implication of each is as follows:

- Concur - Proceed with the project according to the current plan. An example would be where the approver is not aware of any issues for the current stage.

- Concur with qualifications - There are issues or concerns. The project can proceed according to the current plan if an acceptable action plan is developed for each issue by the stage exit meeting. An example would be where there is no plan for testing an interface to an existing system that is being changed.

- Non-concur - There are very significant issues or concerns. The project should not move to the next stage until issue(s) are resolved. An example would be where funding for the project has been withdrawn or not appropriated.

All qualifications (issues/concerns) must be communicated to the project manager. The position response form contains space for this purpose, however other forms of communication may be used.

Responses are not required from individuals in the "Support" or "Information" categories of the distribution list; however, they are encouraged to review the deliverables and provide feedback that may have an impact on the project plan.

Prepare Action Plans: The project manager must prepare an action plan to address each qualification received. Sometimes action plans extend beyond the stage exit milestone. This is acceptable, if it will not negatively impact the current project plan. These action plans are then presented at the stage exit meeting.

Conduct Exit Meeting: At the exit meeting, the project manager presents positions from the approvers, along with qualifications raised during the stage exit process, and issues that remain open from the In-Stage Assessment (ISA). Action plans must also be presented for each qualification or issue. The objective is to demonstrate that all issues have been resolved, the current plan is sound, and the project is under control. The results of the meeting are documented in summary form, and include positions, qualifications, action plans, and follow up activity.

Output: The work products produced when the Stage Exit process is executed consists of the following:

- Positions from the approvers
- Qualifications (if any) from review of the deliverables
- Action plans to resolve all qualifications/issues.

Meeting Outcome: The results of the exit meeting will determine the next step in the development process. The project will proceed in one of the following directions.

- Project proceeds to the next stage according to plan. There were no qualifications raised.

- Project proceeds to the next stage according to plan. All qualifications raised had an acceptable action plan.

- Project cannot proceed to the next stage because significant issues were raised that do not have acceptable action plans to resolve; e.g., funding withdrawn. Schedule a follow-up exit meeting to review action plans and reach concurrence to proceed.

Management Briefings: Periodically, (e.g. quarterly) the quality assurance analyst will brief the senior or upper level manager (e.g. functional, or contract manager if appropriate) regarding the health and well being of the project, from the QA analyst's perspective. This will minimize the possibility of any surprises later and, if issues exhist, they can be addressed timely. The briefing should cover the following areas:

- Project status
- Issue(s) (if any)
- Project risk(s)
- Action(s) required to remove issues or mitigate risk

Prior to briefing the upper level manager, the QA analyst will brief the project manager and the project managers's manager. This should be more as a matter of courtesy; there should be no surprises, since these persons are involved with the project ongoing.

If serious issues exhist, the QA analyst will provide a briefing on an unscheduled basis, rather than wait until the next periodic meeting.

Responsibility Matrix: The following matrix defines the responsibilities of the various parties involved in the Stage Exit process.

	Project Manager (1)	Support Areas (2)	Quality Assurance (QA)	System Owner	User POC	Project Mgr.'s Mgr. (3)	Process Owner/Team		
Schedule Stage Exit	P		R						
Distribute current project plan and other materials	P								
Review project plan etc.		R	R	R	R	R			
Concur/non-concur			P	P	P	P			
Prepare action plans	P								
Conduct exit meeting	P		S						
Monitor process effectiveness			P				S		
Continuous process improvement	S		S			S	P		

P = Perform R = Review S = Support

(1) Project Manager is the generic term for the person responsible for planning and day-to-day control of the project; e.g., task leader, team manager, project leader.

(2) For example, Information Development, Training, Network Planning.

(3) The Project Manager's manager or above.

Example Forms

The following pages provide filled in examples that can be used as a guide for completing the Stage Exit documents.

Example 1
Stage Exit Notification Memo

Date: April 15, 1996

To: Distribution list

From: John Doe, Project Manager

Subject: Requirements Defintion Stage Exit for the New Billing System

The Requirements Definition Stage Exit for the New Billing System (NBS) has been scheduled. The exit meeting will be held on June 1, 1996 from 9:00 a.m. to 11:00 a.m. at the Important Building, room A410. Positions, qualifications, and action plans will be reviewed at this meeting.

The review material is attached. A response is required by May 15, from those persons designated as approvers on the distribution list. All others are encouraged to provide feedback and attend the exit meeting. The approver's response may be a position of concur, concur with qualifications (issues), or non-concur. A position form is attached for your convenience. A non-response has the effect of a concurrence for that approver.

If you have any questions, please contact me at (301) 555-1234 or Mary Smith at (301) 555-1235 for assistance.

Project Manager

Attachment

cc: Distribution list (attached)
Project notebook/file

(memo only)
As appropriate

Example 2
Stage Exit Distribution List

Distribution list:

APPROVAL		Function
A. B. Brown	Client Representative	Concurs that project is ready to proceed into next stage, based on completed deliverables for the current stage, a sound revised project plan for next stage, and no open issues.
C. D. Green	Project Mgr.'s Mgr.	
E. F. Grey	User POC	
G. H. White	Quality Assurance	
A. A. Azur	System Owner	

SUPPORT[3]		Function
I. J. Smith	Documentation	Reviews revised project plan for their functional area involvement and services that project is dependent upon. Provides feedback to the project manager.
K. L. Jones	Network Engineering	
M. N. King	Training	
O. P. Johnson	Security	
Q. R. Other	others as appropriate	

INFORMATION		Function
Q. R. Here	Owner management	Provided for information purposes. Feedback is welcome, but not required.
S. T. There	User management	
U. V. Every	Contractor management	
W. X. Where	others as appropriate	

[3] In the Planning Stage, the distribution list must include all support areas. In subsequent stages, only the support areas involved.

Example 3
Stage Exit Participation - LAN Project

	$))) SYSTEM DEVELOPMENT LIFECYCLE STAGES / EXITS $))▸						
	Planning	Reqrmts	Functl Design	System Design	Progrmg	Soft Intg & Testng	Instaln & Acptn
APPROVERS	SE	SE	SE	SE	SE	SE	SE
System Owner	R	R	R	R	R	R	R
User POC	R	R	R	R	R	R	R
Project Mgr.'s Mgr.	R	R	R	R	R	R	R
Quality Assurance	R	R	R	R	R	R	R
SUPPORT AREAS							
Network Engineering	N	X			X		X
Network Operations	N		X		X		X
Computer Operations	N						
System Programming	N						
Data Base Administration	N						
System Software Administr.	N						
Capacity Planning	N						
Security	N	X					
Documentation	N	X	X	X	X		X
Training	N		X		X	X	X
Records Management	N					X	
Micro Systems Research	N		X				
IRM Planning	N					X	

SE = Stage Exit.
R = Participation is required.
N = Notification. All areas are notified when the first Stage Exit is scheduled.
X = Sample selection.

Note: This is only an example. The project manager must identify those support areas who need to participate in each stage of development for a given project. The actual list of participants may vary from the example provided.

Example 4
Stage Exit Participation - Mainframe Project

	SYSTEM DEVELOPMENT LIFECYCLE STAGES / EXITS						
	Planning	Reqrmts	Functl Design	System Design	Progrmg	Soft Intg & Testng	Instaln & Acptn
APPROVERS	SE	SE	SE	SE	SE	SE	SE
System Owner	R	R	R	R	R	R	R
User POC	R	R	R	R	R	R	R
Project Mgr.'s Mgr.	R	R	R	R	R	R	R
Quality Assurance	R	R	R	R	R	R	R
SUPPORT AREAS							
Network Engineering	N						
Network Operations	N						
Computer Operations	N			X	X	X	X
System Programming	N			X	X		X
Data Base Administration	N	X			X	X	X
System Software Administr.	N	X		X	X	X	
Capacity Planning	N			X	X		
Security	N	X					
Documentation	N	X	X	X	X		X
Training	N		X		X	X	X
Records Management	N					X	
Micro Systems Research	N		X				
IRM Planning	N					X	

SE = Stage Exit.
R = Participation is required.
N = Notification. All areas are notified when the first Stage Exit for a project is scheduled.
X = Sample selection.

Note: This is only an example. The project manager must identify those support areas who need to participate in each stage of development for a given project. The actual list of participants may vary from the example provided.

Example 4A
Stage Exit Participation - Client/Server Project

	$))) SYSTEM DEVELOPMENT LIFECYCLE STAGES / EXITS $))▸						
	Planning	Reqrmts	Fuctnl Design	System Design	Progrmg	Soft Intg & Testng	Instaln & Acptn
APPROVERS	SE	SE	SE	SE	SE	SE	SE
System Owner	R	R	R	R	R	R	R
User POC	R	R	R	R	R	R	R
Project Mgr.'s Mgr.	R	R	R	R	R	R	R
Quality Assurance	R	R	R	R	R	R	R
SUPPORT AREAS							
Network Engineering	N	X			X		X
Network Operations	N		X		X		X
Computer Operations	N	X					
System Programming	N		X				
Data Base Administration	N	X	X	X			
System Software Administr.	N						
Capacity Planning	N		X	X			
Security	N	X					
Documentation	N	X	X	X	X		X
Training	N		X		X	X	X
Records Management	N					X	
Micro Systems Research	N		X				
IRM Planning	N					X	

SE = Stage Exit.
R = Participation is required.
N = Notification. All areas are notified when the first Stage Exit is scheduled.
X = Sample selection.

Note: This is only an example. The project manager must identify those support areas who need to participate in each stage of development for a given project. The actual list of participants may vary from the example provided.

Example 5
Stage Exit Position Response Form

******** TO BE SENT TO APPROVERS ********

Project name: New Billing System (NBS)

Project stage: Analysis

Return form to: John Doe, Project Manager CDSI / CXXI / Room 1234

Return by: May 15, 1996

Position:

☐ Concur.
Proceed with the project according to the current plan.

☐ Concur with qualifications.
Issue(s) exist. The project can proceed according to the current plan if there is an acceptable action plan for each issue by the stage exit meeting.

☐ Non-concur.
Significant issue(s) exist. The project should not proceed to the next stage until the issue(s) is resolved.

Qualifications (issues):

_____

_____

_____

_____

_____

Approver: C. D. Whomever
(please print)

Signed: _____ Date _____

Example 6
Stage Exit Position Response Form

******** RECEIVED FROM AN APPROVER - CONCUR WITH QUALIFICATIONS ********

Project name: New Billing System (NBS)

Project stage: Analysis

Return form to: John Doe, Project Manager CDSI / CXXI / Room 1234

Return by: May 15, 1996

Position:

☐ Concur.
Proceed with the project according to the current plan.

√ ☐ Concur with qualifications.
Issue(s) exist. The project can proceed according to the current plan if there is an acceptable action plan for each issue by the stage exit meeting.

☐ Non-concur.
Significant issue(s) exist. The project should not proceed to the next stage until the issue(s) is resolved.

Qualifications (issues):

1. In the requirements document, there are no requirements for expected response times, for both the first and subsequent screens of each transaction.
2. The maximum concurrent number of users will be 150, rather than 130 as stated in the requirements document.

3. The prerequisite equipment cannot be installed earlier than October 1, 1996. This creates a 2-week variance from the current project plan of record.

Approver: C. D. Whomever
(please print)

Signed: Signature Date April 26, 1996

Example 7
Stage Exit Position Response Form

******** RECEIVED FROM AN APPROVER - NON-CONCUR ********

Project name: New Billing System (NBS)

Project stage: Analysis

Return form to: John Doe, Project Manager CDSI / CXXI / Room 1234

Return by: May 15, 1995

Position:

☐ Concur.
Proceed with the project according to the current plan.

☐ Concur with qualifications.
Issue(s) exist. The project can proceed according to the current plan if there is an acceptable action plan for each issue by the stage exit meeting.

√ ☐ Non-concur.
Significant issue(s) exist. The project should not proceed to the next stage until the issue(s) is resolved.

Qualifications (issues):

1. The proposed technical solution will require extensive retraining of personnel in the branch offices. There is no funding available for this activity this year or in the foreseeable future. A different solution must be explored.

Approver: C. D. Whomever
(please print)

Signed: Signature Date April 26,1994

Templates

The following pages can be copied and used for the Stage Exit documents.

Stage Exit Notification Memo

Date:

To: Distribution list

From: *[project manager]*

Subject: *[stage name]* Stage Exit

The *[stage name]* Stage Exit for *[project name]* has been scheduled. The exit meeting will be held on *[date]*, from *[start time]* to *[end time]*, at *[location including room number]*. Positions, qualifications, and action plans will be reviewed at this meeting.

The review material is *[attached or has been distributed previously]*. A response is required by *[date]* from those persons designated as approvers on the distribution list. All others are encouraged to provide feedback and attend the exit meeting. The approver's response may be a position of concur, concur with qualifications (issues), or non-concur. A position form is attached for your convenience. If no response is received, a position of concur will be assumed for that approver.

If you have any questions, please contact me at *[area code/phone number or electronic address]* or *[alternate name, phone number]* for assistance.

[sign here if hardcopy]

Project Manager

Attachment

cc: Distribution list (attached)
Project notebook/file

(memo only)
[list individuals as appropriate]

Stage Exit Position Response Form

Project name: _____

Project stage: _____

Return form to: _____ _____

Return by: _____

Position:

☐ Concur.

Proceed with the project according to the current plan.

☐ Concur with qualifications.

Issue(s) exist. The project can proceed according to the current plan if there is an acceptable action plan for each issue by the stage exit meeting.

☐ Non-concur.

Significant issue(s) exist. The project should not proceed to the next stage until the issue(s) is resolved.

Qualifications (issues):

_____

_____

_____

_____

_____

_____

Approver: _____
(please print)

Signed: _____ Date _____

Software Engineering Methodology

Appendix F
Computer System Retirement Guidelines

Purpose: This procedure establishes the process for the orderly retirement of information systems regardless of software platform or size, both classified and unclassified. It also addresses the disposition of IS records according to Department of Energy (DOE) requirements listed in the references. It does not apply to the Teleprocessing Services Program (TSP) software. It may not include the removal of information systems from the software libraries used for software reuse and sharing.

These procedures should be followed to remove software, data, and references to an information system from all computer operating platforms and to notify other offices who keep records of computer applications that a system has been retired.

Initiation and distribution: The procedure for *Computer System Retirement Guidelines* is initiated by the Chief Information Officer (CIO) for adaption by all DOE organizations that may be involved in a system retirement.

References: DOE 1324.2A, RECORDS DISPOSITION, September 13, 1988

National Archives and Records Administration (NARA) Handbook on Managing Electronic Records, 1990.

DOE Software Engineering Methodology, March 1996.

General Information: The System Owner or their organization's Proper Program Manager has overall responsibility for retirement of an information system. The federal or contractor System Manager is responsible to the organization's (federal) Information Management (IM) point-of-contact and System Owner for the orderly review and processing of the retirement.

Disposition of Retiring Information Systems: Disposition for a retiring information system will depend on retirement conditions. The System Owner must identify this criteria in the retirement initiation memo, and have it certified by their Proper Program Manager or the DOE or Headquarters (if at Headquarters) Records Management Officer.

Data is not being converted to another data base: If the system is being "totally" retired, the data must be preserved in accordance with DOE 1324.2A, RECORDS DISPOSITION, for that system's type of data and the time period specified in the Order. All software required to manipulate the data must be retained for the same duration. Requests for a "total" retirement must be supported by an approved DOE F 1324.10, Records Inventory and Disposition Schedule (RIDS), and a DOE F 1324.5, Request for Records Disposition Authorization.

Disposition of Retiring Information Systems, continued:

Data is being converted to a newly developed or redesigned data base: If the data will not be retired and archived, but will remain active in a new data base, the recordkeeping requirements will be minimal. Computer resident data shall not be destroyed until all machine-readable data and software (e.g., production source, load modules, job control language) are backed up to tape or other magnetic or optical media, and retained for the specified time period in a approved and secured vault. In addition, a complete copy of all system documentation must be retained with the information system retirement tapes either in the vault, or in a safe area for easy retrieval to support any eventual reactivation or sharing of the system or software.

Identification of Retiring Information Systems:

The identification of information system retirement candidates can occur in several ways.

- System Owners may identify systems that are no longer needed or are being replaced by a new system.
- Through the annual long-range planning process by Information Management points-of-contact.
- Capacity planning/performance management reports may show systems are no longer needed through indications of inactivity such as low computing resource, computer time, and disk space utilization.
- An Information System Review or other operating system audit process may reveal inactive systems.
- The System Manager may identify no longer needed systems through a semiannual review for an assigned organization. In these cases, such as information systems with low maintenance activity, the System Manager may recommend that the System Owner initiate a retirement request.

In all cases, it is the System Owner's responsibility to request and authorize the retirement process.

Retirement Process:

Several phases constitute the retirement process. The System Owner coordinates the retirement with their organization's Records Liaison Officer (RLO) and Proper Program Manager, site Records Management Officer, DOE Departmental or Headquarters Records Officer, and authorizes the

Retirement Process, continued:

system retirement process through their Information Management Division. The System Manager coordinates the process through the System Owner's Support Programmer Staff. The Support Programmer oversees the retirement process at the direction of the System Manager to ensure each participant completes their part of the retirement process. Each participant notifies the Support Programmer when their actions have been completed. Upon completion of all tasks, a "retirement folder" is forwarded to the System Manager who prepares a memorandum for signature by the Director of Information Management or Operations (whichever is appropriate) to the System Owner that the retirement has been completed. These phases are discussed in more detail in the Retirement Procedure.

Definitions:

Computer System Engineering Services: The team or organization responsible for the systematic approach to the specification, design, development, testing, operation, and maintenance of a computer system.

Computer/Microcomputer/Local Area Network Client-Server Operations: The organization responsible for the maintenance of mainframe, minicomputer, microcomputer, local area network client/server functions including continuity of operations, operating system, and application backups.

Director of Information Management: The manager with responsibility for the planning, management, and control of information as a corporate asset.

Director of Operations: The manager with the overall responsibility for the management of the functional operations of information systems.

Proper Program Manager: The manager with the responsibility for a program area and the information system providing automated computing for the program area's missions and functions. The authorizing signature on the Records Inventory and Disposition Schedule.

Quality Assurance Team: The team that conducts software quality assurance activities, such as reviewing plans and deliverables for compliance with applicable standards, and providing guidance and assistance on process matters.

Records Liaison Officer: The office employee who provides recordkeeping assistance to the Office and is the Office point-of-contact dealing with records management issues with the site Records Management Officer.

Definitions, continued:

Records Management Officer: The site authority for records management. Directly communicates with the Departmental Records Officer at Headquarters.

Support Documentation/Technical Writing: The organization responsible for assisting with the development of system documentation during a software engineering project, and maintaining current documentation through subsequent versions.

Software Library/Inventory Coordinator: The organization or individual with the responsibility of maintaining site software libraries, repositories, and inventories.

System Support Programmer: The support programmer with knowledge of the system being retired sufficient to perform tasks assigned in the retirement process.

System Review Inventory System (SRIS) Coordinator: The employee with the responsibility of reporting new systems in development and the status of existing systems through the annual UNICALL planning process.

System Owner: The information or program manager's system owner of record who is responsible for all development, enhancement, and maintenance activates associated with the system.

System Manager: Has the overall responsibility to ensure the support programmer coordinates and completes the procedures in this guideline with all participants in the retirement process.

Participants: The following is a list of participants involved in the retirement process.

Participant	Responsibility
System Owner	Initiates the retirement process via memo to the Director of Information Management or Operations.
Site Records Management Officer or Records Liaison Officer, and Proper Program Manager	Provides recordkeeping guidance and assistance in preparing DOE F 1324.10, Records Inventory and Disposition Schedule (RIDS); and DOE F 1324.5, Request for Records Disposition Authorization, and authorizes disposition.
System Manager	Provides direction to Support Programmer in the retirement process.
Support Programmer	Coordinates retirement process with retirement participants at the System Manager's direction. Conducts all software and data backup.
Computer Systems Engineering Services/Operations	Removes system from automated scheduling, continuity of operations, backup program, and software subsystems.
System Documentation or Technical Writing	Coordinates retirement of system documentation. Sends one copy to support programmer to store with the retirement backup tapes.
Software Library/Inventory Coordinator	Maintains library for all computer applications.
System Review Inventory System (SRIS) Coordinator	Maintains the SRIS for the retired information systems.
Director of Information Management or Operations (whichever is appropriate)	Reviews retirement request and forwards to System Manager. Originates and signs the retirement completion memorandum.

Initiate Retirement:

This section provides guidelines for initiating the retirement procedure, which begins with the System Owner and records management officials. In the context of this document, the System Owner has the overall responsibility for system retirements. However, it may be appropriate for System Managers, Local Area Network Managers, or organizational Information Management points-of-contact to initiate a system retirement process.

Participant	Step	Action
System Owner	1	Coordinate retirement with the organization's Proper Program Manager, RLO, and DOE Departmental and/or Headquarters Records Manager. Submit an approved DOE F 1324.10, Records Inventory and Disposition Schedule (shown in figure 1), and DOE F 1324.5, Request for Records Disposition Authorization (shown in figure 2) in accordance with DOE 1324.2A, as appropriate.
	2	Prepare a memorandum authorizing the retirement (shown in figure 3).
	3	Submit DOE F 1450.6, Request for System Control Identification Number (shown in figure 4), to request that any system control identification numbers be deleted, if necessary.
	4	Send the memorandum and copies of DOE F's 1324.10, 1324.5, and 1450.6 to the Director of Information Management or Operations. Send copies to memorandum cc: list.
Records Management Officer	5	Provide recordkeeping guidance as appropriate, and assistance in preparing DOE F 1324.10 and 1324.5.

Figure 1. DOE F 1324.10, Records Inventory and Disposition Schedule

Note: Figure is not available in electronic form in this document.

Figure 2. DOE F 1324.5, Request for Records Disposition Authorization

Note: Figure is not available in electronic form in this document.

[Date]

[System Owner Routing Symbol]

Retirement of *[system name; acronym; and module number, if applicable)]*

Director of Information Management (or Information Resource Management)/Operations

[Depending upon the circumstances, choose one of the following two paragraphs for the first paragraph]

This is your authorization for total retirement of the *[system name]*, effective *[date]*. We no longer use the system and do not require access to the data. The data and software may be archived according to the time periods specified in the attached DOE F 1324.5. The system is operated on a *[state hardware platform--standalone microcomputer, Local Area Network, VM processor, MVS processor.*

OR

This is your authorization for the partial retirement of *[system name]*, effective *[date]*. We no longer use the system and the data has been converted to another/new data base. The data will remain active at the same classification level in the new data base and should not be retired or archived. The system is operated on a *[state hardware platform--standalone microcomputer, Local Area Network, VM processor, MVS processor.*

[The following is the second paragraph of the retirement memorandum.]

This retirement process has been coordinated with my organization's Records Liaison Officer and the Proper Program Manager, and the Records Management Officer. I, therefore, authorize you to archive and retain the programs, files (and data when appropriate), and documentation for the time specified in the attached DOE F 1324.10, Records Inventory and Disposition Schedule (RIDS), so that the system may be reactivated during that period, if necessary. If the system is not reactivated by the end of the retention period, dispose of it according to regular procedures.

[System Owner]
[Title]
[Organization]

Attachments
DOE F 1424.5
DOE F 1324.10
DOE F 1450.6, as appropriate

cc:
Organization's Proper Program Manager
System Manager
Computer System Engineering Services
Records Management Officer
DOE Departmental Records Officer (for DOE-wide systems)

Figure 3. Memorandum Authorizing Retirement

Figure 4. DOE F 1450.6, Request for System Control Identification Number

Note: Figure is not available in electronic form in this document.

Review the Authorization Request:

Participant	Step	Action
Director of Information Management or Operation (whichever is appropriate)	1	Review retirement request and forward it to the appropriate System Manager.
System Manager	2	Review retirement authorization and determine if it addresses all requirements of this procedure and that it is acceptable. **If acceptable ...** Provide Support Programmer with copies of the document and direction for processing the retirement. **If not acceptable ...** Discuss concerns with the System Owner and resolve conflicts.
	3	Place the original authorization documents in the system project file.
	4	Issue a revision to the support work order, if appropriate.
	5	Provide direction to the Support Programmer for coordinating retirement.

Review the Authorization Request, continued:

Participant	Step	Action
Support Programmer	1	Coordinate retirement at the direction of the System Manager.
	2	Establish a system retirement folder and prepare a retirement checklist (shown in figure 5) and a schedule.
	3	Prepare a system retirement notification memorandum (shown in figure 6). Send memo to the specified organizations and related support staff.
	4	Execute the removal of the system using the retirement checklist.
	5	Monitor the retirement actions of each of the organizations outlined in the retirement notification memorandum.
	6	Prepare a final SRIS form and send to Chief Information Officer.
	7	Submit DOE F 1450.6, Request for System Control Identification Number, to Computer System Engineering Services/Operations.

Retirement Checklist

- ☐ Retirement authorization letter from the system owner. (*)
- ☐ Support work plan signed off (one system work plan) or work plan revised (multi-system work plan) per System Manager direction. (*)
- ☐ Memorandum to all specified organizations and support areas to notify them of retirement. (*)
- ☐ DOE F 1450.6, Request for System Control Identification Number, completed by System Owner for deletion of system module number, if applicable. (*)
- ☐ Final SRIS form completed and sent to Chief Information Officer. (*)
- ☐ Copy of signed DOE F 1324.10 (RIDS) placed in retirement folder, if appropriate. (*)
- ☐ For a "total" system retirement, a copy of DOE F 1324.5, Request for Records Disposition Authorization, must be approved and signed by the site Records Management Officer. (*)
- ☐ Final report cycle offered to System Owner.
- ☐ Production source, load modules, job control language, and all other system (information system, operational, etc.) software copied to tape (or diskette for microcomputer), and placed in a secured vault.
- ☐ One complete copy of system documentation (User System Reference Manual and Programmer System Reference Manual) stored with tapes or easily accessible area.
- ☐ Remaining system documentation removed from all sources and Documentation/Technical Writing Departments.
- ☐ Modules deleted from Test Source and Test Load Libraries.
- ☐ Modules deleted from the Production Source and Load Libraries.
- ☐ PROCS deleted.
- ☐ Revoke all access (LOGONIDs/USERIDs) to the system.
- ☐ Coordinated retirement with Software Library/Inventory Coordinators.
- ☐ Submit DOE F 1450.6, Request for System Control Identification Number, to Computer Systems Engineering Services/Operations, if applicable. (*)
- ☐ Prepare proposed retirement completion memorandum from the Director of Information Management or Director of Operations (whichever is appropriate) to the System Owner. (*)
- ☐ Other (specify)_____.

***** Copy to be included in the retirement folder.

Figure 5. Retirement Checklist

Memo to: *[Distribution List]*

From: *[Support Programmer Manager/Supervisor]*

Subject: Notification of Intent to Retire the *[system name, (acronym), system module number, if applicable]*

This is to inform all concerned organizations that the *[system name, (acronym), system module number]*, is being retired as of *[date]*. Attached is the memorandum received from the System Owner that authorized this retirement (Attachment 1). Also, included is a list of the activities that were coordinated by the Support Programmer for the retirement process (Attachment 2).

The primary purpose of this system was *[state purpose]*. *["Note that this system was used to process classified data" or "contained data which has been determined to be privacy data."]* This system was designed by *[system designer]* and installed on *[date]*. This system was in *[production/development/process of implementation]* status when *[System Owner]* of *[Organization and Location]* requested retirement for *[reason of retirement]*. The approximate annual operating cost, including computer processing, programming and user support in the last year before retirement was $ *[obtain from the Division of Information Technology Services; if the total is less that $100, enter "less than $100"]*.

Additional actions required by associated organizations to retire this system are as follows:

- Computer Operations - Follow DOE F 1324.10, Records Inventory and Distribution Schedule (RIDS), or DOE F 1324.5, Request for Records Disposition Authorization, for defining the disposition of all tapes associated with the system. Ensure retirement and archive tapes, and a complete set of system documentation is placed in a secure vault for the specified period, after which they may be disposed of without further notice. A copy of the software may be left in the software library/inventory for sharing/reuse by other organizations.

- Computer System Engineering Services/Operation

  - Remove all production JCL from the automatic scheduling programs and continuity of operations (Vital Records Protection) program, if applicable. Remove from CICS, ORACLE, OMEGA, BasisPlus, etc., if applicable. Revoke all access to the system on the effective date: *[list of access/USERIDs]*.

  - Follow DOE F 1324.10, Records Inventory and Distribution Schedule (RIDS), or DOE F 1324.5, Request for Records Disposition Authorization, for defining the disposition of all tapes associated with the system. Ensure retirement and archive tapes, and a complete set of system documentation is placed in a secure vault for the specified period, after which they may be disposed of without any further notice.

- Quality Assurance/Audits - Remove the system from inventories of classified, sensitive and tracking systems, and scheduled information system and security reviews. A review may be conducted 6 months after system retirement to assure all retirement procedures have been followed.

- Support Documentation/Technical Writing - Notify the operations and maintenance support staffs, System Owner and all holders on the distribution list to destroy their copies of system documentation due to retirement. Remove all stored documentation (excluding software libraries/inventories). Send one complete copy of the system documentation for storage with the retirement tapes or in easily accessible area.

Thank you for your cooperation in the retirement of the system. Please contact the *[Support Programmer name]* at *XXX-XXXX* when your organization has completed assigned retirement procedures.

Attachments

Distribution List
Organization's Proper Program Manager
System Program Manger
Records Management Officer
Director of Information Management
Director of Operations
Computer System Engineering Services
System Manager
Quality Assurance
Computer Security Site Manager
Software Library/Inventory Coordinator
SRIS Coordinator
Support Documentation/Technical Writing

Figure 6. System Retirement Notification Memorandum

SAMPLE HEADQUARTERS SYSTEM RETIREMENT ACTION MEMORANDUM

ACTIONS COMPLETED FOR *[SYSTEM NAME (ACRONYM)]* RETIREMENT

By *[Support Programmer Name]*

- All program listings and other miscellaneous listings were destroyed.
- The following disk files and software were copied to tape (005621):

DOE.D0413LIB.LOAD to D0413.S002.LOAD	File #1
DOE.DOE413LIB.CNTL to D0413.S003.CNTL	File #2
DOE.D0413LIB.CLIST to D0413.S004.CLIST	File #3
LIBRARIAN (SOURCE) PROD470 to D0412.S005.COBOL	File #4
and D0413.S006.COBOL	File #5
DOE.D413201.DATA to D0413.S007.DATA	File #6

NOTE: This activity freed up 103 tracks on DASD.

- The *[SYSTEM ACRONYM]* data base has been copied to two tapes:

010216, 011316, DSN=D0413.S001.DBBKUP

NOTE: This activity freed up 323 tracks on DOE 8XX.

- All tape files previously used by this system have been released.
- One complete copy of system documentation was stored with retirement tapes.
- All other copies of documentation were removed and destroyed.
- System Module No. *XXXX* can be removed from the System Module Cross-Reference System (SMXREF).
- All source programs have been deleted from LIBRARIAN.

Figure 6. Attachment - Sample Headquarters System Retirement Action Memorandum

Remove from Software Platform:

Participant	Step	Action
Computer System Engineering Services	1	Remove the system from software subsystems (CICS, OMEGA, etc.) if appropriate.
	2	Delete the system access rules.
	3	Delete the Timesharing Option (TSO) account number, if appropriate. If a user has access to this system only, cancel the TSO LOGONID.
	4	Delete the system JCL and dispose of all associated documentation and files, as appropriate.
	5	Remove the system from automatic scheduling.
	6	Remove the system from the Continuity of Operations Program schedule.
	7	Assign tape retention periods as specified on DOE F 1324.10 or DOE F 1324.5. Deliver tapes to a secure vault, and documentation to the same vault or a secure, easily accessible area.
	8	Inform Support Programmer of the action(s) taken.

System Documentation:

Participant	Step	Action
Support Documentation and/or Technical Writing	1	Send one complete copy of the system documentation to the Support Programmer for storage with the retirement backup tapes.
	2	Remove stored system documentation (excluding software library/inventory).
	3	Inform all holders of retiring system documentation to destroy their copies.
	4	Inform Support Programmer of the action(s) taken.

Assemble the Retirement Folder:

Participant	Step	Action
Support Programmer	1	Assemble the system retirement folder as follows: • System Retirement Checklist • System Owner's retirement authorization memorandum • Signed support work plan • DOE F 1450.6, Request for System Control Identification Number • Final SRIS form • Approved DOE F 1324.10, Records Inventory and Disposition Schedule (RIDS) *or for total system retirement* Approved DOE F 1324.5, Request for Records Disposition Authorization • Proposed retirement completion memorandum from Director of Information Management or Operations to the System Owner

Assemble the Retirement Folder, continued:

Participant	Step	Action
Support Programmer	2	Task the retirement process. **If the system is tasked under separate order...** Copy of Support Work Order signed in the "Completion" space. **If the system is tasked as part of a multi-system work order...** Identify the need to revise the work plan and an estimate of the revised work plan for the remaining systems.
	3	Forward the folder with the retirement completion memorandum (shown in figure 7) to the System Manager.

[Date]

HR-4

Retirement of *[system name (system acronym); module number, if applicable]*

[System Owner]

In accordance with your request, dated *[date]*, the system retirement of the *[system name]*, has been completed effective *[date]*.

All software, procedures, programs and data have been copied to tape. All system access has been revoked. The retirement tapes and one complete copy of system documentation has been stored in a secure vault, and will be retained as indicated in DOE F 1324.10, Records Inventory and Disposition Schedule (RIDS). Upon expiration of the retention period, tapes and documentation will be destroyed without further notice. All computer resident data has been deleted from system libraries.

[name], Director
Information Management/Operations

Attachments

cc:
Director of Information Management, or
Director of Operations
Organization's Proper Program Manager
System Manager
Records Management Officer
Computer Security Site Manager
System Manager
Computer System Engineering Services
DOE Departmental Records Officer (for DOE-wide systems)

Figure 7. Retirement Completion Memorandum

Review the Retirement Folder:

Participant	Step	Action
System Manager	1	Review the retirement folder for completeness and accuracy.
	2	Resolve any existing conflicts concerning the system retirement with the System Owner.
	3	Finalize the retirement completion memorandum (shown in figure 7) for the Director of Information Management or Director of Operations signature. Forward the retirement completion memorandum to the Director for signature.

Approval of System Retirement:

Participant	Step	Action
Director of Information Management or Operations	1	Approve or disapprove the system retirement. **If approved ...** Sign the retirement completion memorandum and transmit to System Owner appropriate carbon copies as indicated in figure 7. **If disapproved ...** Return the retirement memorandum to the System Manager for resolution.
System Manager	2	Finalize the retirement process. **If approved ...** Go to *Final System Retirement Tasks*. **If disapproved ...** Resolve problems that prompted disapproval. Revise the retirement completion memorandum and return it to the Director for approval.

Final System Retirement Tasks:

Participant	Step	Action
System Manager	1	Close out system project file with information from the system retirement folder, including the retirement completion memorandum (figure 7) from the Director of Information Management or Director of Operations.
	2	Request SRIS Coordinator to delete system from SRIS inventory.
	3	Request the software library/inventory coordinator to delete system from libraries, if appropriate.
Software Library and/or Inventory Coordinator	4	Review the retirement memorandum and delete the system from the library, if appropriate. The coordinator may leave the system in the inventory for software reuse and sharing. If retained, post a disclaimer with the software stating that it is being offered for sharing and reuse, but is unsupported as of the date of retirement. Retain the retirement memorandum.
SRIS Coordinator	5	Remove system record from SRIS.