System Development and Life-Cycle Management (SDLCM) Methodology

Handbook, Version 2.2

Release Date: December 1999

Prepared Under TAC: C90046

Prepared by

COMPUTER SCIENCES CORPORATION
Federal Sector – Civil Group
15245 Shady Grove Road
Rockville, Maryland 20850

Prepared for

CISSCO Program
Applications Development Division
Office of the Chief Information Officer
U.S. Nuclear Regulatory Commission
Washington, D.C. 20555-0001

Under

FEDSIM Contract Number: GS00K96AJD0012
Task Order: K0096AJ9611

System Development and Life-Cycle Management (SDLCM) Methodology Handbook, Version 2.2

TAC C90046

December 1999

Approved by:

_____ _____

M. J. Bassman — Date
CSC CISSCO Lead Technical Reviewer

_____ _____

M. E. Ellwood — Date
CSC CISSCO Quality Assurance Manager

_____ _____

M. J. Bassman — Date
CSC CISSCO Senior Information Engineer

_____ _____

T. C. Sullivan — Date
CSC CISSCO Program Director

System Development and Life-Cycle Management (SDLCM) Methodology

Handbook, Version 2.2

December 1999

UNITED STATES NUCLEAR REGULATORY COMMISSION
WASHINGTON, DC

Foreword

Nuclear Regulatory Commission (NRC) Management Directive 2.5, "Application Systems Life-Cycle Management," establishes the policies for applications systems life-cycle management. The System Development and Life-Cycle Management (SDLCM) Methodology implements Directive 2.5 by providing life-cycle structure and guidance to NRC projects.

The SDLCM Methodology comprises seven components:

1. Define Initial Project Requirements
2. Acquire Support Resources
3. Design the Solution
4. Engineer the Solution
5. Deploy the Solution
6. Service the Solution
7. Decommission the Solution

The methodology is *not* itself a document or a set of documents. It is *the approach to doing business* at NRC, and it is described by a set of documents, including but not limited to the following:

- *SDLCM Methodology Handbook*
- *SDLCM Methodology Procedures, Standards, and Forms*
- *SDLCM Methodology Tool Inventory*
- *SDLCM Methodology Overview Training*

Table of Contents

Figures

Tables

1. Introduction

1.1 Background

Within the Nuclear Regulatory Commission (NRC), every project aims to provide its customer with an application system product that is engineered to satisfy the customer's requirements, within determined cost, schedule, and quality guidelines. The ability to provide such a product is made easier when the project team follows a comprehensive and consistent methodology. The System Development and Life-Cycle Management (SDLCM) Methodology provides life-cycle structure and guidance to NRC projects.

1.2 Objectives

The objectives of this *SDLCM Methodology Handbook* are three-fold. First, this handbook defines the SDLCM Methodology application *system* life cycle and describes the component structure and each of the seven components. It also relates the component structure to the *software* development life cycle, which is embedded within two of the components.

Second, this handbook introduces some system and software engineering terminology and provides a common basis of understanding for all users of the methodology.

Finally, this handbook identifies each of the life-cycle roles that must be performed by management and technical personnel and clarifies the responsibilities of each of those roles.

1.3 Scope

This handbook discusses the SDLCM Methodology. It includes the development and the life-cycle management (that is, maintenance and enhancement) of NRC application systems from the definition of the initial project requirements (after a project has been identified) through the decommissioning of a system that is no longer to be used.

It does *not* include strategic technology and business systems planning.

Further, the SDLCM Methodology does not apply to the development and maintenance of NRC's internal network and communications services infrastructure, which supports the applications systems.

1.4 Overview

Chapter 2 of this handbook presents an overview of the SDLCM Methodology. A careful reading of that chapter is recommended. The remainder of this document can be treated like a reference book. Chapter 3 provides guidance to a project manager on selecting an appropriate software life-cycle model. Chapters 4 and 5 introduce the support activities of quality assurance (QA) and configuration management (CM), respectively. The next seven chapters describe in detail the seven components of the SDLCM Methodology.

Appendix A explains how to propose changes to the SDLCM Methodology or to the documentation set (including this handbook) that describes the methodology. Appendix B summarizes the roles introduced in this handbook and the responsibilities of the personnel who

perform those roles. Appendix C discusses the required products of a project that develops, enhances, or maintains a system using the SDLCM Methodology. Appendix D includes a summary list of all activities discussed in this handbook. Appendix E summarizes the process for transition of legacy systems from NRC to a contractor for life-cycle management. Appendix F is a glossary of important terms.

2. Methodology Overview

2.1 A Structured Approach

The SDLCM Methodology is a structured approach to designing, developing, deploying, maintaining, and decommissioning information systems. It addresses all aspects of an information systems solution from cradle to grave. It allows, and even encourages, flexibility within a clearly defined structure.

This handbook presents processes that clearly and unambiguously define what needs to be done, by whom, when, why, and how:

- Who? ⇒ Roles and their responsibilities
- What? ⇒ Activities and sub-activities
- When? ⇒ Sequence as illustrated in the diagrams
- Why? ⇒ Products
- How? ⇒ Tools and Techniques

Everyone involved in the application system development and maintenance process at NRC is required to use the SDLCM Methodology. Users include:

- Customers who are developing and maintaining their own systems with the involvement of the Office of the Chief Information Officer (OCIO)
- Contractors who are supporting customer- or OCIO-developed systems
- All OCIO personnel

2.1.1 Seven Components

Figure 2–1 illustrates the component structure of the SDLCM Methodology. Seven components are specified:

1. **Define Initial Project Requirements.**
   After the need for a project has been established, identify the information management problem, clarify the scope, identify functional and data requirements, analyze alternative solutions, select appropriate tools, establish a project plan, and develop a support resource request.
2. **Acquire Support Resources.**
   Obtain the necessary resources (for example, staff, technology, contractors, training) to ensure timely and effective progress on the project.
3. **Design the Solution.**
   Analyze the (functional, data, communications, network, and interface) requirements, design the system solution, and prepare integration and project plans.
4. **Engineer the Solution.**
   Acquire or construct all modules of the system, test the system, create the rollout strategy, and prepare the training materials.

5. **Deploy the Solution.**
   Install and roll out the integrated solution.
6. **Service the Solution.**
   Maintain or enhance an existing application system or its support environment.
7. **Decommission the Solution.**
   Inactivate and cease support of an application system.

Strategic technology and business systems planning activities (shown outside the broken line delineating the seven methodology components) are conducted prior to the start of Component 1. Those activities are outside the scope of the SDLCM Methodology and are not discussed in this handbook.

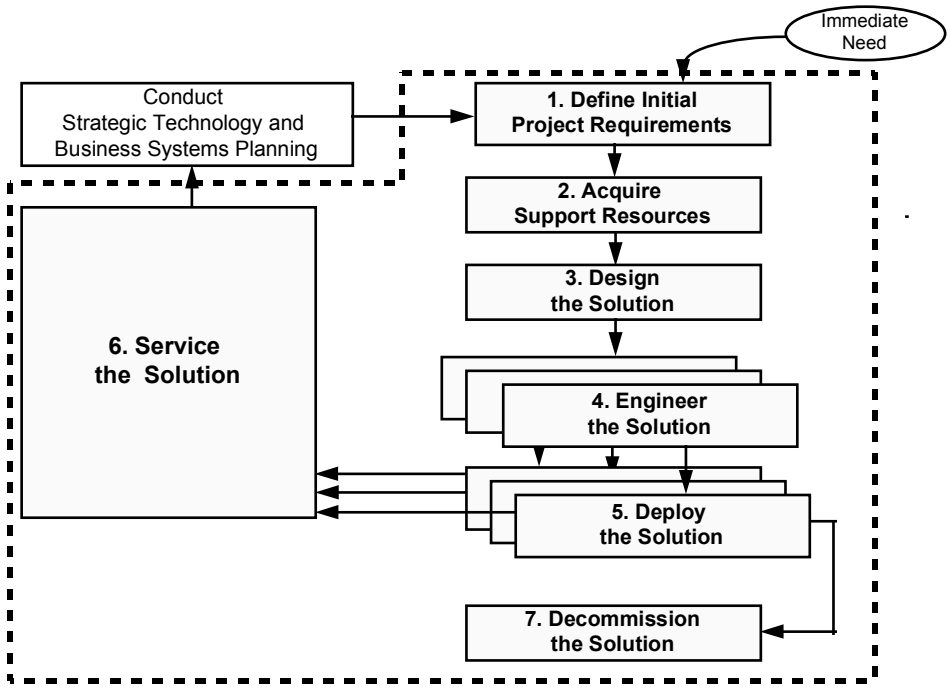


Figure 2–1. Component Structure

2.1.2 Presentation of the Components

Subsequent chapters of this handbook describe each of the components in detail. The chapter for each component includes:

- A brief summary of the purpose of the component
- A table of roles and responsibilities
- Identification of the entry criteria, that is, the events that may trigger the activities within the component
- A list of inputs to the components
- A data flow diagram illustrating the overall flow of the activities performed within the component
- Identification of the outputs of the component

- A suggested list of techniques to support the performance of the activities and a pointer to a list of tools
- Identification of the exit criteria, that is, the conditions that are expected when the component is complete
- A more detailed description of each of the activities shown in the data flow diagram

2.2 Principles and Assumptions

The following *principles* are incorporated throughout the methodology:

- Strategic technology and business systems planning is conducted prior to the start of the Define Initial Project Requirement component of the methodology.
- When laws and mandates require specific activities to be completed, these activities are incorporated into the methodology at the appropriate time.
- All requests (new applications *and* upgrade or maintenance requests on existing applications) will be initiated through the first component, Define Initial Project Requirements.
- All systems requests will be evaluated for Business Process Reengineering (BPR) in the first component *before* proceeding to the other components within the methodology.
- OCIO will evaluate the end-user desktop environment and work with the user to resolve compatibility issues.
- The SDLCM Methodology (including both the process descriptions and the inventory of tools) will be reviewed and revised as required to support the evolving needs of NRC and also to implement improvements suggested by lessons learned from applying the methodology on NRC projects.

The following *assumptions* are in place:

- Contractors may fill any of the project roles *except* Executive Sponsor and Business Advocate.
- Systems analysts will be trained in facilitation, methodology, and specific tools as needed.
- The BPR techniques used will be standardized and will use automated tools that will be contained in the inventory.
- There will generally be a uniform environment with some variations permitted and managed as necessary.

2.3 Activities and Products

This section explains the terms used in Figure 2–2.

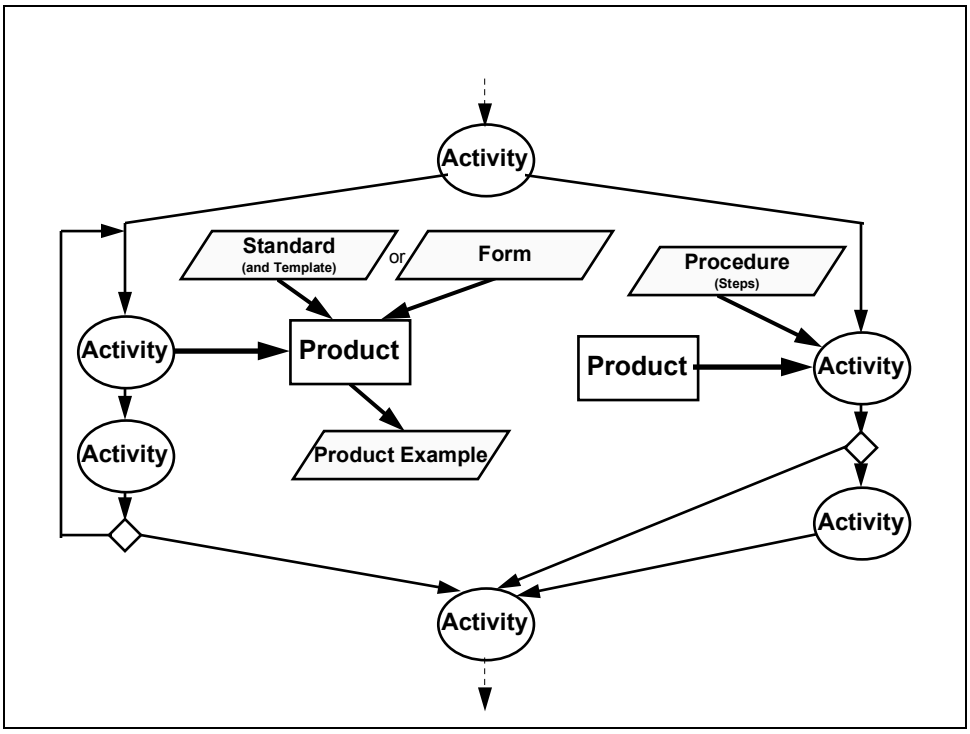


Figure 2–2. Relationships between Activities and Products

Each of the seven components of the methodology comprises a collection of *activities*. As suggested by the figure, the activities are not necessarily performed sequentially. Some of the activities may be performed in parallel as illustrated by the two separate paths branching from the activity shown at the top of the figure. The activities in the left branch illustrate that some iteration may be required. The activity on the lower right of the figure is performed optionally depending on some characteristic of a particular project.

A *procedure* is a written description of the roles, responsibilities, and steps required for performing a complex activity or a subset of an activity. Within a procedure, *steps* are performed sequentially. Each SDLCM Methodology procedure is identified by the prefix "P–" followed by a four-digit number. If an activity is described completely in this handbook or in a companion guidebook, then no separate procedure is provided.

A *product* is software or associated information created, modified, or incorporated to satisfy the project requirements. Examples include plans, requirements, design, code, databases, test information, and manuals. A product is an output of an activity and may be input to a subsequent activity.

A *standard* is a written set of criteria used to develop and evaluate a product or to provide and evaluate a service. A product standard (for example, the standard for a Project Action Plan) includes an annotated outline of the product. A non-product standard (for example, the standard for data models) documents a common form or approach (data models, for example, are required in several product standards). For each product standard, a word processor *template* is provided to facilitate the production of a product. Each SDLCM Methodology standard is identified by the prefix "S–" followed by a four-digit number.

A *form*, rather than a product standard, is provided when the resulting product can be produced simply by filling in a set of blanks or completing a set of questions. Each SDLCM Methodology form is identified by the prefix "F–" followed by a four-digit number.

A *product example* is an instance of a product developed by members of a project team using a standard or form. Representative products are provided as examples to make it easier for future project teams to satisfy the product requirements.

The specific project products required by this methodology are identified in subsequent chapters and summarized in Appendix C. Product examples are maintained in the system documentation library. Procedures, standards, and forms are provided in a separate volume (see *SDLCM Methodology Procedures, Standards, and Forms*) that supports this handbook.

2.4 Feedback and Improvement

Every project provides an opportunity to improve the process defined by the SDLCM Methodology. Figure 2–3 illustrates the QA and process improvement feedback loops for projects.

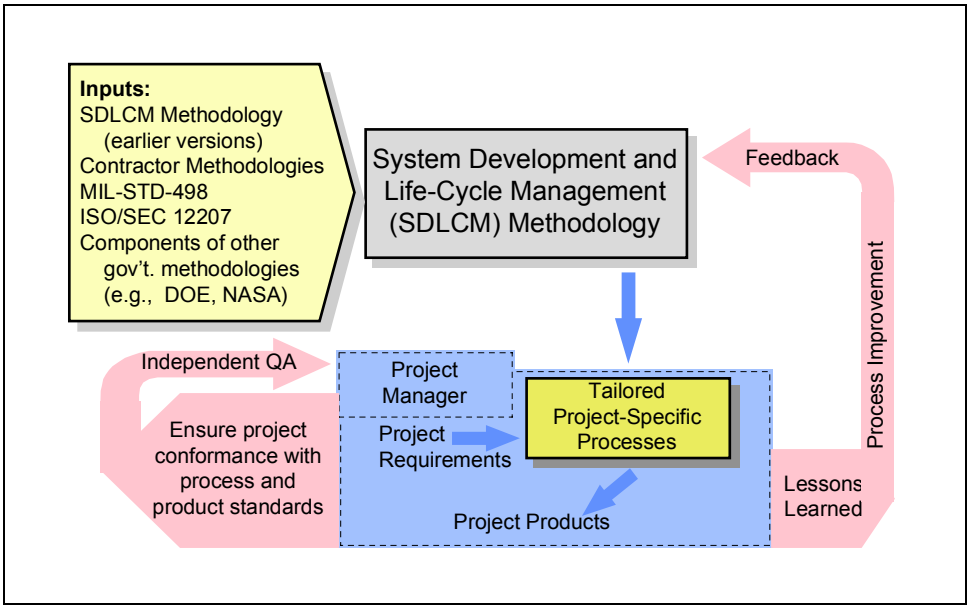


Figure 2–3. A Living Methodology

The SDLCM Methodology was originally based on the experiences of NRC and its contractors from earlier projects and on understanding gained from studying other similar methodologies.

When a new project is initiated, the project manager reviews the methodology in the context of the project requirements and derives a tailored project-specific process, which is reviewed and approved by a QA representative. As the project team applies the process to the development of products, the QA representative reviews the progress to ensure that both the process and the products conform to the approved standards. Any inconsistencies are reported to the Project Manager. If the Project Manager fails to take appropriate action, the independent QA organization elevates the report to higher levels of management. The QA process is defined in more detail in Chapter 4.

Throughout the life of a project, and especially as a part of project closeout activities, lessons learned from applying the SDLCM Methodology to the projects are fed back to the SDLCM Methodology Team for analysis. It is primarily the actual project experience of managers and team members that sustains the process improvement feedback loop shown in the figure. The mechanism for reporting methodology deficiencies or suggesting improvements is described in Appendix A, Maintaining the SDLCM Methodology.

2.5 Systems Concepts

Figure 2–4 illustrates the concept of a system life cycle. This section clarifies some terms related to the system life cycle.

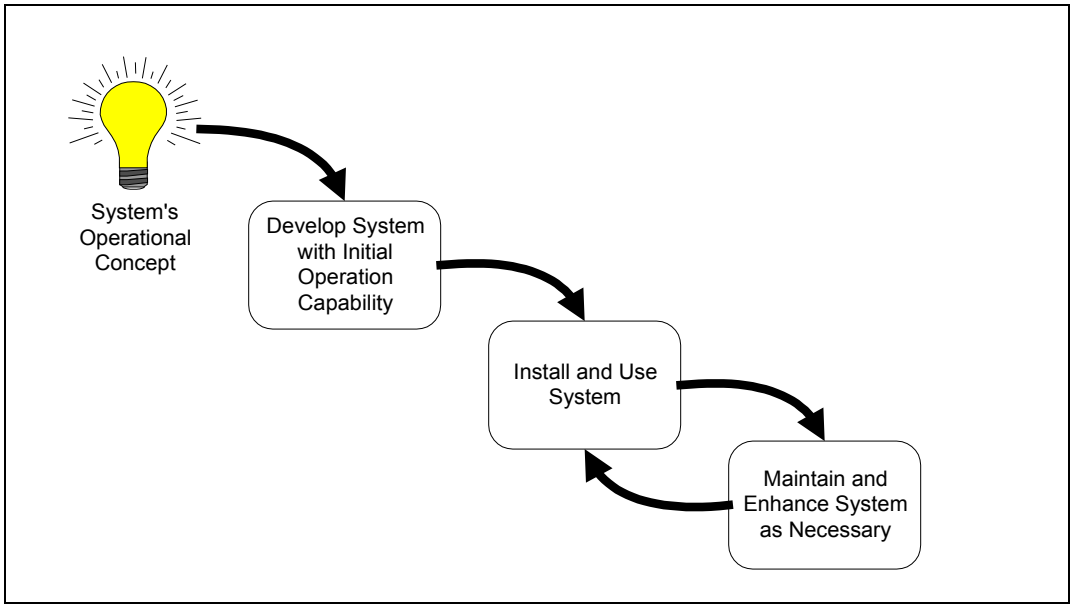


Figure 2–4. Illustrative System Life Cycle

In this handbook, the term *system* (or *application system* or *information system*) refers to the operational entity that the NRC organization is responsible for developing, maintaining, or enhancing. That is, if the organization is responsible for developing several software and hardware configuration items (CIs) *and* is responsible for integrating them into an operational entity, then the collection of those CIs is the system. If, however, the organization is responsible for developing a single software CI, which may be integrated into (for example) a management information system by a different NRC organization, then the software CI itself is the system referred to in this handbook.

Some of the systems that NRC organizations develop, or maintain and enhance, include multiple CIs. There may be a mix of software CIs and hardware CIs, or the system may be only hardware or only software. This guidebook discusses the activities associated with the development and with the maintenance and enhancement of a system. When the system comprises only software CIs, then most of the system- and CI-level products are one and the same.

Development (see Figure 2–5) is the creation and installation of an operational system that meets an initially defined set of system requirements. Once the system is operational, subsequent changes are considered *maintenance* or *enhancement* (see Figure 2–6). Many of the maintenance

and enhancement activities are the same or similar to those used in development. The SDLCM Methodology process applies equally to development and to maintenance and enhancement efforts.

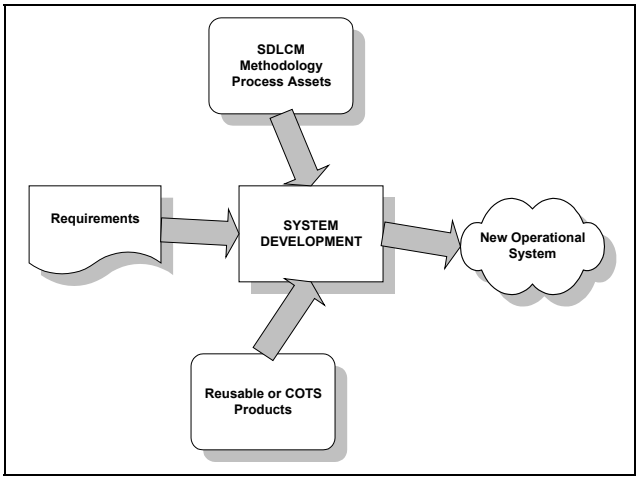


Figure 2–5. Development Context

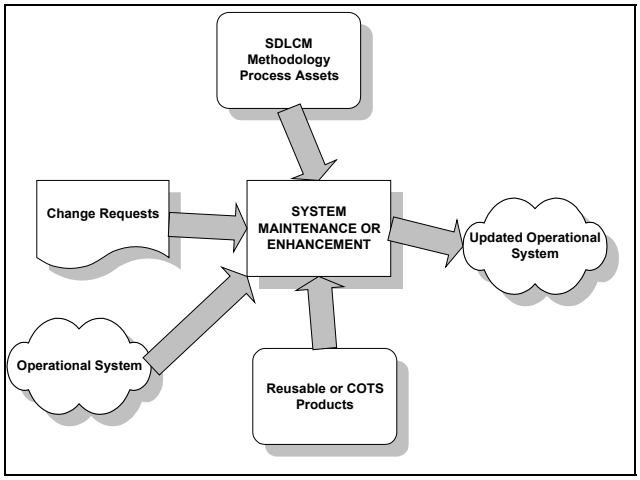


Figure 2–6. Maintenance or Enhancement Context

3. Selecting an Appropriate Software Development Life-Cycle Model

The software development life cycle falls within the systems life cycle introduced in the preceding chapter. It also falls within Component 3 (Design the Solution) and Component 4 (Engineer the Solution) of the SDLCM Methodology. This chapter introduces four software development life-cycle models that may be applied to software development projects at the NRC. Software enhancement projects typically follow the same life-cycle model used for the original development activities. The maintenance life cycle (Component 6) is discussed in Chapter 11.

A life-cycle model comprises one or more phases (for example, a requirements definition phase, a design phase, a test phase). Each phase is defined as the time interval between two scheduled events. For example, in the waterfall life-cycle model, the design phase is defined as the period between the software specification review (SSR) and the critical design review (CDR).

Within each phase, one or more activities are executed. For example, during the waterfall model's design phase, the design activity is performed; the test planning activity may be done at the same time. In most cases, activities neither begin nor end precisely at the phase boundaries; rather, they overlap adjacent phases, as illustrated in Figure 3–1.

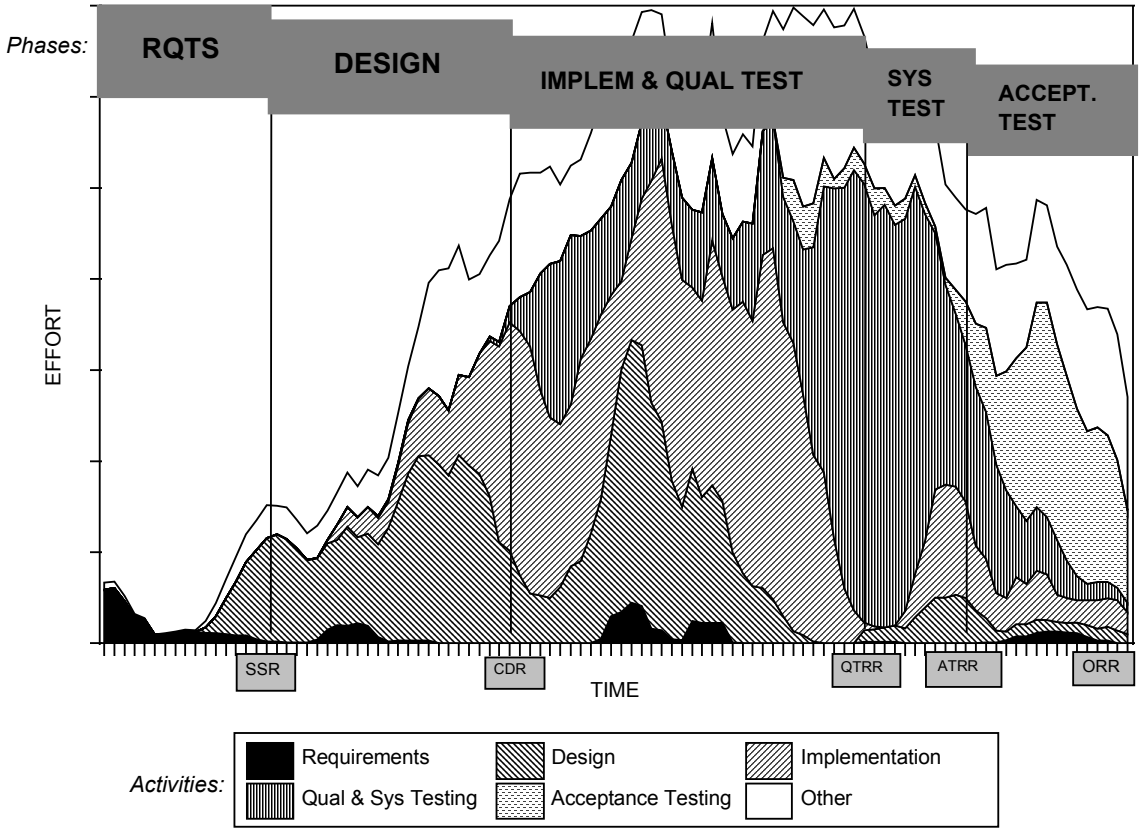


Figure 3–1. Phases and Activities

Notice that life-cycle phases are defined by time boundaries, whereas activities are defined by the type of work being performed.

Various methods (or techniques) may be used in the performance of an activity. For example, object-oriented design is one proven design method; structured design is another.

This handbook does not mandate any particular software life-cycle model, and the order of activities described here is not intended to conform to any particular model. Few specific methods are mandated for required activities. These decisions are left to the manager, who selects an appropriate life-cycle model and activity-related methods and defines them in the Project Action Plan. This chapter contains guidance on selecting an appropriate, recommended life-cycle model and methods for many activities.

For convenience, Table 3–1 provides the definitions (see the Glossary) of several important terms used extensively in this chapter.

Table 3–1. Defining a Life Cycle

Term	Definition
Software life cycle	"The period of time that begins when a software product is conceived and ends when the software is no longer available for use." [IEEE 610.12] A life cycle is typically divided into life-cycle phases.
Life-cycle model	A framework on which to map activities, methods, standards, procedures, tools, products, and services (for example, waterfall, and spiral).
Life-cycle phase	A division of the software effort into non-overlapping time periods. Life-cycle phases are important reference points for the software manager. Multiple activities may be performed in a life-cycle phase; an activity may span multiple phases.
Activity	A unit of work that has well-defined entry and exit criteria. Activities can usually be broken into discrete steps.
Method	A technique or approach, possibly supported by procedures and standards, that establishes a way of performing activities and arriving at a desired result.

Four life-cycle models are summarized in the following subsections:

- Waterfall development life-cycle model
- Incremental development life-cycle model
- Evolutionary development life-cycle model
- Package-based development life-cycle model

3.1 Waterfall Development Life-Cycle Model

Table 3–2 summarizes the life cycle defined by the waterfall development model.

Table 3–2. Summary of Waterfall Development Life-Cycle Model

Summary description and discussion	The waterfall (single-build) life-cycle model is essentially a once-through-do-each-step-once approach. Simplistically, determine user needs, define requirements, design the system, implement the system, test, fix, and deliver the system. This model is illustrated in Figure 3–2.
Advantages	• Well-studied, well-understood, and well-defined • Easy to model and understand • Easy to plan and monitor • Many management tools exist to support this life-cycle model
Disadvantages	• Most if not all requirements must be known up front • Does not readily accommodate requirements changes • Product is not available for initial use until the project is nearly done
Most appropriate when ...	• Project is similar to one done successfully before • Requirements are quite stable and well-understood • The design and technology are proven and mature • Total project duration is relatively short (less than a year) • Customer does not need any interim releases

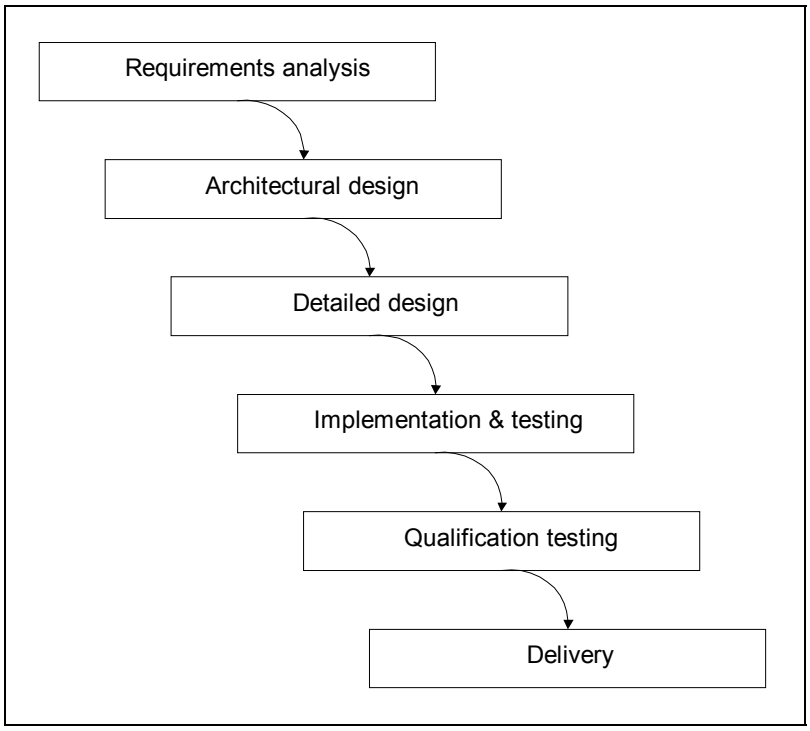


Figure 3–2. Waterfall Development Life-Cycle Model

3.2 Incremental Development Life-Cycle Model

Table 3–3 summarizes the life cycle defined by the incremental development model.

Table 3–3. Summary of Incremental Development Life-Cycle Model

Summary description and discussion	The incremental (multi-build) life-cycle model determines user needs and defines a subset of the system requirements, then performs the rest of the development in a sequence of builds. The first build incorporates part of the planned capabilities; the next build adds more capabilities; and so on, until the system is complete. This model is illustrated in Figure 3–3.
Advantages	• Reduces risks of schedule slips, requirements changes, and acceptance problems • Increases manageability • Interim builds of the product facilitate feeding back changes in subsequent builds • Interim builds may be delivered before the final version is done; this allows end users to identify needed changes • Breaks up development for long lead time projects • Allows users to validate the product as it is developed • Allows software team to defer development of less well understood requirements to later releases after issues have been resolved • Allows for early operational training on interim versions of the product • Allows for validation of operational procedures early • Includes well-defined checkpoints with customer and users via reviews
Disadvantages	• Like the waterfall life-cycle model, most if not all requirements must be known up front • Sensitive to how specific builds are selected • Places products (particularly requirements) under configuration control early in the life cycle, thereby requiring formal change control procedures that may increase overhead, particularly if requirements are unstable
Most appropriate when ...	• Project is similar to one done successfully before • Most of the requirements are stable and well-understood; but some TBDs may exist • The design and technology are proven and mature • Total project duration is greater than one year or customer needs interim release(s)

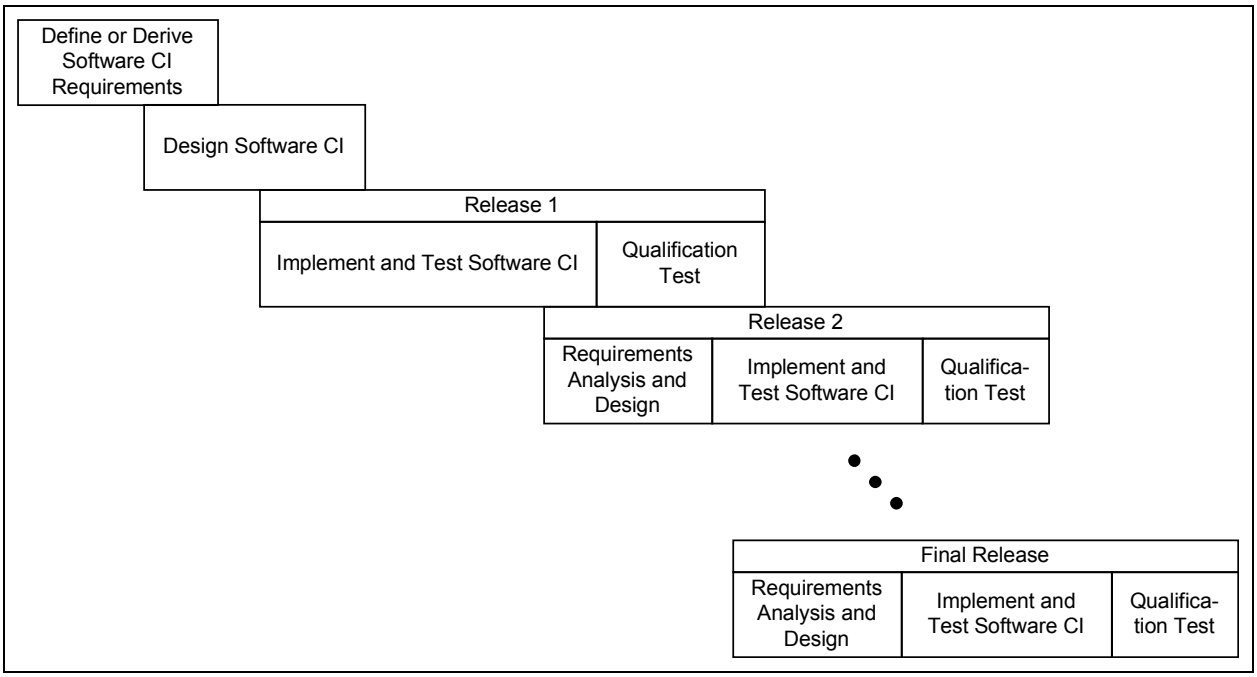


Figure 3–3. Incremental Development Life-Cycle Model

3.3 Evolutionary Development Life-Cycle Model

Table 3–4 summarizes the life cycle defined by the evolutionary development model.

Table 3–4. Summary of Evolutionary Development Life-Cycle Model

Summary description and discussion	Like the incremental development model, the evolutionary life-cycle model also develops a system in builds, but differs from the incremental model in acknowledging that the user needs are not fully understood and not all requirements can be defined up front. In the evolutionary approach, user needs and system requirements are partially defined up front, then are refined in each succeeding build. The system evolves as the understanding of user needs and the resolution of issues occurs. Prototyping is especially useful in this life-cycle model. (The evolutionary development life-cycle model is sometimes referred to as a spiral development model, but it is not the same as Boehm's spiral model. This model is also sometimes referred to as a prototyping life-cycle model, but it should not be confused with the prototyping technique. This life-cycle model is illustrated in Figure 3–4.
Advantages	• Not all requirements need be known up front • Addressing high risk issues (for example, new technologies or unclear requirements) early may reduce risk • Like the incremental life-cycle model, interim builds of the product facilitate feeding back changes in subsequent builds • Users are actively involved in definition and evaluation of the system • Prototyping techniques enable developers to demonstrate functionality to users with minimal of effort • Even if time or money runs out, some amount of operational capability is available
Disadvantages	• Because not all requirements are well understood up front, the total effort involved in the project is difficult to estimate early. Therefore, expect accurate estimates only for the next cycle, not for the entire development effort. • Less experience on how to manage (progress is difficult to measure) • Risk of never-ending evolution (for example, continual "gold plating") • May be difficult to manage when cost ceilings or fixed delivery dates are specified • Will not be successful without user involvement
Most appropriate when ...	• Requirements or design are not well-defined, not well-understood, or likely to undergo significant changes • New or unproved technologies are being introduced • System capabilities can be demonstrated for evaluation by users • There are diverse user groups with potentially conflicting needs

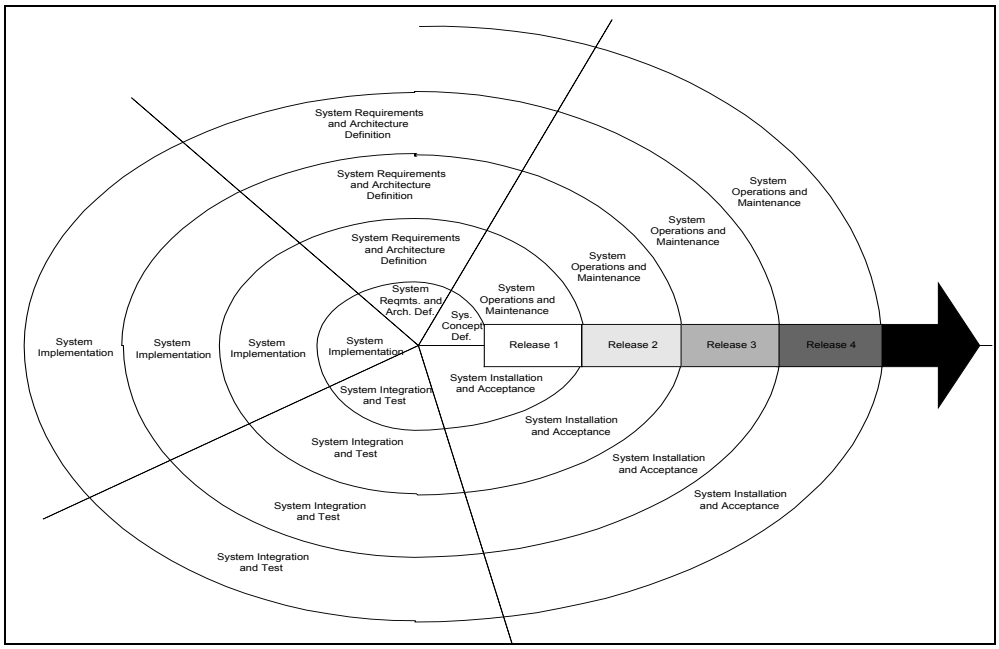


Figure 3–4. Evolutionary Development Life-Cycle Model

3.4 Package-Based Development Life-Cycle Model

Table 3–5 summarizes the life cycle defined by the package-based development model.

Table 3–5. Summary of Package-Based Development Life-Cycle Model

Summary description and discussion	The package-based development life-cycle model is used for system development based largely on the use of commercial-off-the-shelf and Government off-the-shelf products and reusable packages. Typically, some custom software development is needed to provide interfaces among the non-developed items (NDIs). This model is illustrated in Figure 3–5.
Advantages	• Lower cost than developing equivalent functionality from scratch • Cycle time also often lower than developing equivalent functionality from scratch • Improves confidence in quality of the end product (since quality of NDIs is already known)
Disadvantages	• May result in compromises between desired functionality and functionality provided by NDIs • Maintainability may be more of a challenge because source of NDIs may not be the same NRC organization (for example, requires third party to make changes, raises software CM issues when NDI vendor releases updated versions)
Most appropriate when ...	• A significant portion of the functionality of a system can be provided by NDIs

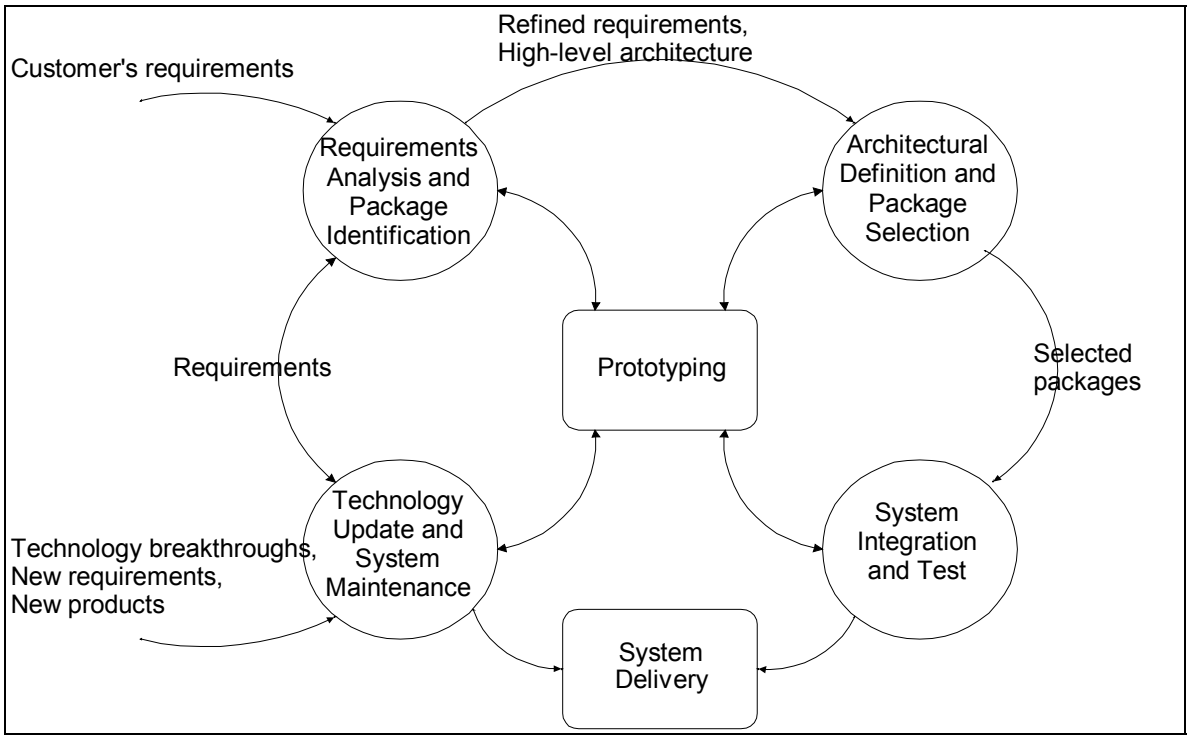


Figure 3–5. Package-Based Development Life-Cycle Model

4. Quality Assurance

4.1 Purpose

Quality Assurance (QA) is a planned and systematic set of activities whose purpose is to provide management with an independent view that approved processes are being used and that high quality products are being produced by NRC project teams.

Quality Assurance involves:

- Reviewing and auditing the activities and products to verify that they comply with published SDLCM Methodology procedures and standards
- Providing managers and project teams managers with the results of these reviews and audits

These review and audit activities occur throughout the life cycles of projects and provide management with visibility needed to control the adherence to established plans, procedures, and standards.

4.2 Quality Assurance Goals

The goals of NRC's Quality Assurance Program are to ensure that:

- The Quality Assurance activities are planned.
- The products produced and activities performed at the NRC adhere to SDLCM Methodology procedures and standards.
- The results of any quality assurance activities are reported to the appropriate groups in a timely manner.
- Any noncompliance issues are corrected at the lowest possible level of management.

4.3 Process Implementation

Each organization, within NRC or under contract to NRC, that develops or maintains software and is subject to the SDLCM Methodology, will prepare and implement a Quality Assurance Plan (QAP). The QAP describes the organization's tailored approach to QA. It identifies the group(s) responsible for performing QA and the relationships between QA and other parts of the organization (such as Program Management, Configuration Management, and the software development and maintenance teams). The plan identifies resources and includes a schedule for performing the required activities. Table 4–1 lists some important QA activities.

Table 4–1. QA Activities

QA Activities	QA Sub-activities
Process Assurance Cycle (PAC) Activities:	Document Project Approach
	Pre-component Process Review
	Component Orientation Meeting
	In-Progress Process Audit
	Life-Cycle Phase Audit
Product Inspection and Certification	
Life-Cycle Model Phase Review	
Document Review	
Audits:	Configuration Audit
	Document Pre-Delivery Audit
	System Product Delivery Audit
	Inspection and Certification Meeting and Materials Audit
	Other In-Process Audits
Non-conformance Reporting and Corrective Action	
Quality Management Activities	
Independent Test Monitoring	
Training Monitoring	
Data Collection and Analysis Monitoring	
QA Activity Reports:	Weekly Reports to management
	Bi-weekly Reports to the organizational Director of Quality Assurance
QA Calendar and Activity Schedules	
Input to Monthly Report	
PAC Related Reports	
Formal and Internal Review Report:	Formal Review
	Formal Review Report
	Internal Review
	Internal Review Report
Audit Reports:	Life-cycle Phase Audit Report
	In-Progress Audit Report
QA Records:	QA Task Files
	Task Area Development Files
	Task Area Maintenance Files

4.4 Measurement and Analysis

Measuring the activities of the QA program permits management to evaluate the proficiency of the QA organization, which leads to better and more efficient planning of QA activities. Measurements may include but are not limited to:

- Completion of milestones for QA activities compared to a given project schedule.
- Work completed, effort expended, and funds expended in the QA activities compared to the planned work.
- Number of process and product reviews and audits compared to the planned work.

4.5 Verifying Implementation

Organizational management holds periodic reviews of the QA program. These reviews are designed to provide management with awareness of and insight into current QA activities.

5. Configuration Management

Configuration Management (CM) is a discipline of applying administrative and technical procedures throughout the software life cycle to:

- Identify, define, and baseline software and associated documentation in a system(s)
- Control modifications to and releases of the baseline
- Record and report the status of the baselines and modification requests
- Ensure baseline completeness, consistency and correctness
- Control storage, handling, and delivery

CM includes the following major activities:

- CM process implementation—definition and documentation of the configuration management activities
- Configuration identification—definition and identification of items subject to configuration control
- Configuration control—evaluation, coordination, and approval or disapproval of proposed changes to controlled items
- Configuration status accounting—recording and monitoring of changes to controlled items
- Data management—maintenance of official correspondence records, CM records, and controlled documentation
- Configuration evaluation—verification that controlled items meet their assigned requirements and are accurately documented

5.1 CM Process Implementation

Each organization, within NRC or under contract to NRC, that develops or maintains software and is subject to the SDLCM Methodology, is required to prepare and implement a Configuration Management Plan (CMP). The CMP describes the organization's tailored approach to CM in conformance with the SDLCM Methodology. It identifies

- The group(s) responsible for performing CM—such as the Configuration Management Office (CMO), Configuration Control Boards (CCBs), and Project CM representatives
- The relationships between CM and other parts of the organization—such as Program Management, Quality Assurance, and the software development and maintenance teams

The plan identifies resources and includes a schedule for performing the required activities.

5.2 Configuration Identification

Configuration identification consists of selecting the configuration items (CIs) for a system and documenting their functional and physical characteristics. It provides the basis for applying management controls to a system configuration, permitting the isolation of items to be controlled, the tracking of their status, and the reporting of their configurations. The CMO, in

cooperation with systems engineering, establishes a scheme for the identification of software configuration items and their versions to be controlled for the project.

For legacy systems, configuration identification begins with the documentation that established the baseline and its version, a complete software inventory of items requiring conversion or transition, associated hardware, and communications components (as required).

For new development, configuration identification begins with requirements that provide the work definition.

Configuration identification also applies to acquired hardware, communications equipment, Commercial-Off-The-Shelf (COTS) software, and documentation. A thorough configuration identification system is essential to monitoring, tracking, and implementing changes.

CI selection involves grouping system components into a unit subject to CM controls. This activity is performed by the systems engineering group, with approval from the appropriate Configuration Control Board (CCB). Certain functional, performance, and physical characteristics are allocated to each CI. A CI that is too large results in decreased visibility and ineffective control; one that is too small or placed under configuration control too early produces the opposite result: visibility that is at a very detailed level for the defined component and control that is excessive and therefore inefficient. For effective CM, CIs must be defined carefully and placed under configuration control at the appropriate time.

The following are guidelines by which CIs should be selected:

- Small and well-defined set of interfaces with other CIs
- Single source, such as a development group or contractor, responsible for providing the CI
- Largest entity that provides adequate client and management visibility into the development process
- Common schedule for acquisition, development, testing, and delivery of subordinate elements
- Independent operational or user interfaces from the user's point of view
- Critical item with respect to safety, security, reliability, or other factors that require specific management attention
- As a single entity, requires maintenance, training, and logistical support
- Specific requirements for performance controls
- High degree of change or modification expected once the CI becomes operational

Because each CI requires its own baseline documentation and configuration control, identification of too many CIs must be avoided. If too many CIs are identified, system development will be excessively controlled and CM will be more complex and costly than necessary.

5.3 Configuration Control

Configuration control is the process of evaluating, approving or disapproving, and monitoring the implementation of changes to baselines during development, operation, or maintenance. This function includes the following:

1. Identification and recording of change proposals (SDLCM Methodology Form F–2502)
   - Approval or disapproval of the request
   - Implementation
   - Verification
   - Release of the modified software item
2. Establishing an audit trail that identifies
   - Each modification
   - Reason for the modification
   - Authorization of the modification

The CMO controls and audits all access to the controlled software items. Activities that handle safety or security critical functions requiring special handling are controlled in accordance with established NRC guidelines.

5.3.1 Responsibility and Process Flow

Configuration control is the responsibility of both the CMO and the CCB.

The CMO is responsible for maintaining the integrity of all baselines. The CMO ensures that only NRC-approved changes are incorporated into baselines. CM-controlled baselines are updated using change control procedures that provide for systematic evaluation, coordination, and formal disposition of proposed changes and for implementation of changes that are approved.

The CCB is the heart of the configuration management process. The CCB provides a structured review process of requirements changes, problem or risk areas, and work in progress. The CCB is established under the authority and control of the Program Management Office and functions in concordance with the CM program. The Program Director, or a designated alternate, chairs the CCB, which is made up of technical managers, a QA representative, a CM representative (who serves as secretary), and other subject matter experts as required for a proper evaluation.

See SDLCM Methodology procedure P–2502, Change Proposal, for a detailed description of the change process and the steps that must be followed to propose and incorporate a change.

See SDLCM Methodology procedure P–2501, Configuration Control Board, for a detailed description of the steps in the CCB process.

5.3.2 Change Vehicles

Changes may be requested using several different vehicles as determined by the type of change. Note that some projects may not require all of these mechanisms, and others may need additional vehicles. The following types of change requests are provided:

- Change Proposal (CP). A formal request to change an approved baseline (software, hardware, equipment, or their specification). See SDLCM Methodology procedure P–2502, Change Proposal, and form F–2502, Change Proposal Form, for additional information.
- Request for Deviation or Waiver. A formal request to deviate from a standard methodology process or to be granted a waiver from following a standard process. In the

context of CM the deviation or waiver typically requests a departure from established configuration baseline requirements. This type of request is discussed further in the chapter on Quality Assurance. See SDLCM Methodology procedure P–2010, Deviations and Waivers, for additional information.

- Problem Report (PR). A formal report of a problem detected during development or maintenance of authorized systems. See SDLCM Methodology forms F–2251 (Problem Reports) and F–2252 (Problem Report Logs).

5.3.3 Change Management

The management of proposed changes to a baseline is a critical part of system maintenance and, hence, of Component 6, Service the Solution, of the SDLCM Methodology. Therefore, the details of this element of configuration management are included in Chapter 11, Section 11.2.

5.3.4 Release Management

The CMO formally controls the release and delivery of software products and documentation. Only certified software and builds are formally delivered and installed. Master copies of code and documentation are maintained for the life of the software product. Replaced or decommissioned functions are archived.

Planning and managing what changes go into each release are critical parts of system maintenance and, hence, of Component 6, Service the Solution, of the SDLCM Methodology. Therefore, the details of this element of configuration management are included in Chapter 11, Section 11.3.

5.4 Configuration Status Accounting

Configuration status accounting is the recording, monitoring, and reporting of all changes to established baselines. The objectives of configuration status accounting are to ensure that:

- All approved changes are reflected in the affected baselines
- No baseline includes changes that have not been approved

The CMO generates management records and status reports that show the status and history of controlled items including software baselines. Status reports include the number of changes for a project, latest versions, release identifiers, number of releases, and the differences between the releases.

5.5 Data Management

Data management (DM) is closely related to configuration status accounting. While configuration status accounting focuses on recording and monitoring changes to controlled items, DM focuses on maintaining official correspondence records, CM records, and controlled documentation.

5.6 Configuration Evaluation

Configuration evaluation involves a series of inspections, tests, reviews, and audits to ensure that the software and associated documentation accurately represent the approved configuration baselines.

The functional completeness of the software items against their requirements is determined using the Functional Configuration Audit (FCA). The FCA is the formal examination of functional characteristics of a configuration item, prior to acceptance, to verify that the item has achieved the requirements specified in its functional and allocated configuration documentation.

The physical completeness of the software items (that is, whether the design and code reflect an up-to-date technical description) are determined using the Physical Configuration Audit (PCA). The PCA is the formal examination of the as-built configuration of a CI against its technical documentation to establish or verify the CI's product baseline.

6. Component 1. Define Initial Project Requirements

6.1 Purpose

The purpose of this component is to:

- Identify an information management problem
- Clarify the scope
- Identify initial functional and data requirements
- Analyze alternative solutions
- Select appropriate tools
- Establish a project plan
- Develop a support resource request
- Secure a Go decision for the project

6.2 Roles and Responsibilities

The roles required to perform the activities in the Define Initial Project Requirements component are shown in Table 6–1.

Table 6–1. Component 1 Roles and Responsibilities

Roles	Responsibilities
Business Advocate	Accountable for Execution
Executive Sponsor	Approval to Proceed
Overall Project Manager Business Project Manager Technical Project Manager Business Subject Matter Expert (SME) Technical SME Other Representatives or Stakeholders • Quality Assurance (QA) Manager • Configuration Management (CM) Manager • Records Management (RM)	Provides Input

6.3 Entry Criteria

Any of the following events may trigger the initiation of Component 1:

- Proactive business planning (includes BPR)
- Proactive technology planning

- Identification of ad hoc business-driven requirements
- Identification of ad hoc technology-driven requirements

Note that these events occur as part of the "Conduct Strategic Technology and Business Systems Planning" box in Figure 2–1, and therefore, are outside the scope of the SDLCM Methodology and this handbook. Any of these triggers may be driven by changes in:

- Cost
- Technology
- External mandate (for example, laws)

6.4 Input, Activities, and Outputs

The inputs to this component are as follows:

- Project Description (new and maintenance)
- Approved budget allocation
- Advance Procurement Plan
- Office or Regional approval
- Responsible project contact
- Project responsibilities
- Technical Reference Model
- Enterprise Model
- Tool Inventory
- Any available Business Process Reengineering (BPR) documents

Figure 6–1 illustrates the flow of the following activities:

1. Clearly Identify Information Management Problem
2. Clarify Technical Scope
3. Notify Records Management Branch to Review Records Management Requirements
4. Identify Functional and Data Requirements
5. Analyze Alternatives
6. Establish a Project Plan
7. Review Toolkit
8. Develop Support Resource Request
9. Plan for Deployment
10. Review (Component 1)

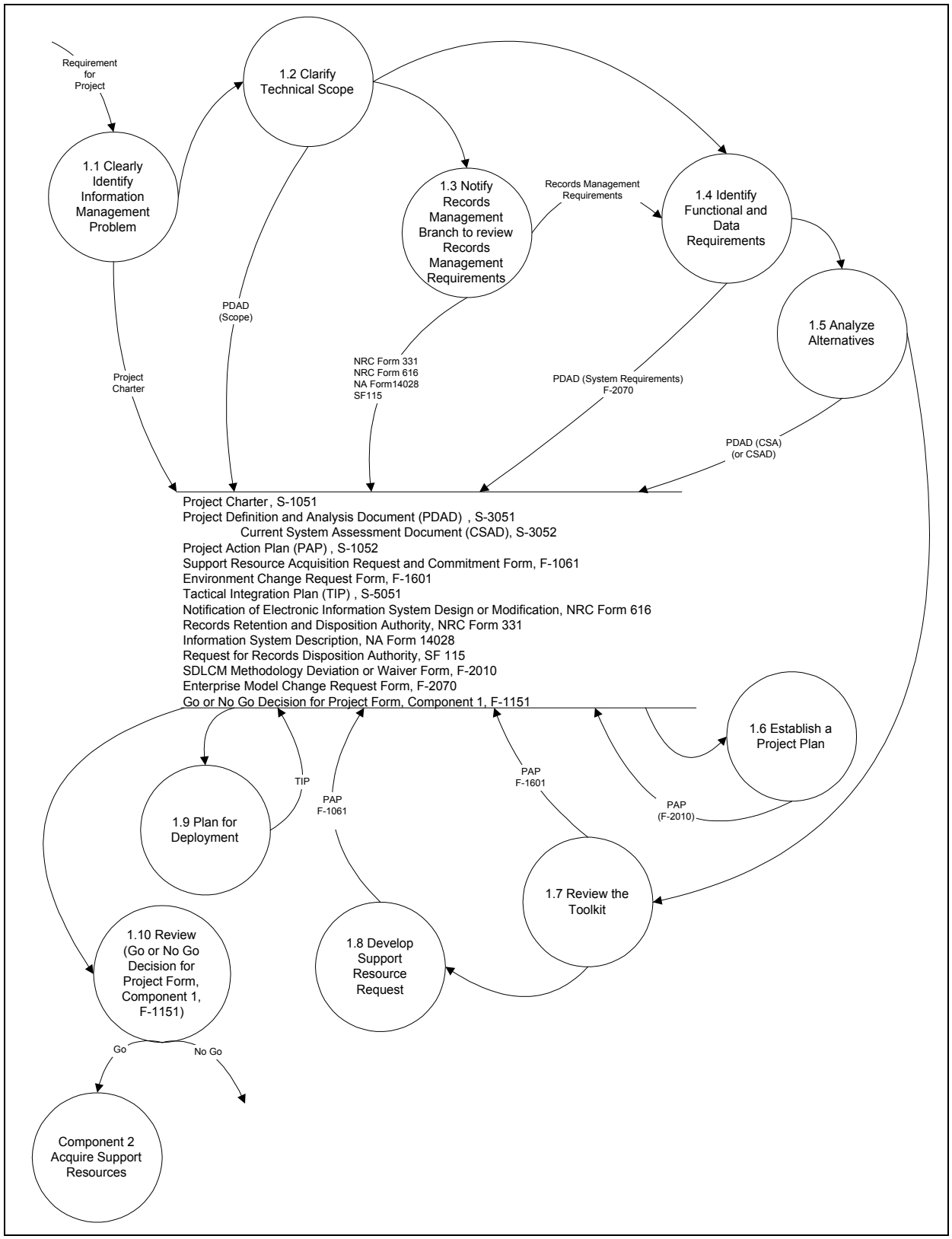


Figure 6–1. Component 1

As the figure suggests, many of the activities can be performed in parallel. For example, Activity 1.6, Establish a Project Plan, may be started as soon as there is enough information to begin planning; the activity cannot be completed, however, until all required planning information has been provided (for example, all system requirements must be known and all tool requirements must be determined).

The activities are described in more detail in Section 6.7.

Figure 6–1 also summarizes the outputs of all Component 1 activities in the central data store.

Appendix C includes a summary description of all products of NRC projects, indicates within which components each product is created and updated, and specifies which products are required. The Current System Assessment Document (CSAD), for example, is not required if there is no current system to be assessed or if the assessment is relatively minor and can be documented adequately in the applicable section of the Project Definition and Analysis Document (PDAD). The PDAD, Project Action Plan (PAP), and Tactical Integration Plan (TIP) are all begun as part of activities within this component and are all updated in subsequent components.

The products, whose standard and form numbers are shown in the figure, are described in more detail in the companion volume *SDLCM Methodology Procedures, Standards, and Forms*.

6.5 Techniques and Tools

The activities of Component 1 are supported by the following techniques.

- Business Area Analysis
- Business Process Reengineering
- Data Modeling
- Process Modeling
- Structured Walkthrough
- Peer Review

Refer to the current *SDLCM Methodology Tool Inventory* for the recommended set of tools.

6.6 Exit Criteria

Component 1 is complete when:

- All major products have been reviewed (structured walkthrough or peer review)
- All major products have been inspected by QA and certified to conform with project standards
- All products have been placed under CM control
- Information has been shared
- Issues have been resolved
- There is management buy-in to move forward
- There are no reasons to stop this project now

6.7 Component 1 Activity Details

The following pages provide detailed activities of Component 1, Define Initial Project Requirements.

Activity 1.1: Clearly Identify Information Management Problem

See Figure 6–2. Using the outputs from the Strategic Technology and Business Systems Planning activities (which are outside the scope of this methodology),

1. Determine what business processes and organizations are within the scope of this project.
2. Determine the business goals, objectives, or problems to be achieved.
3. Summarize the findings in the Project Charter.

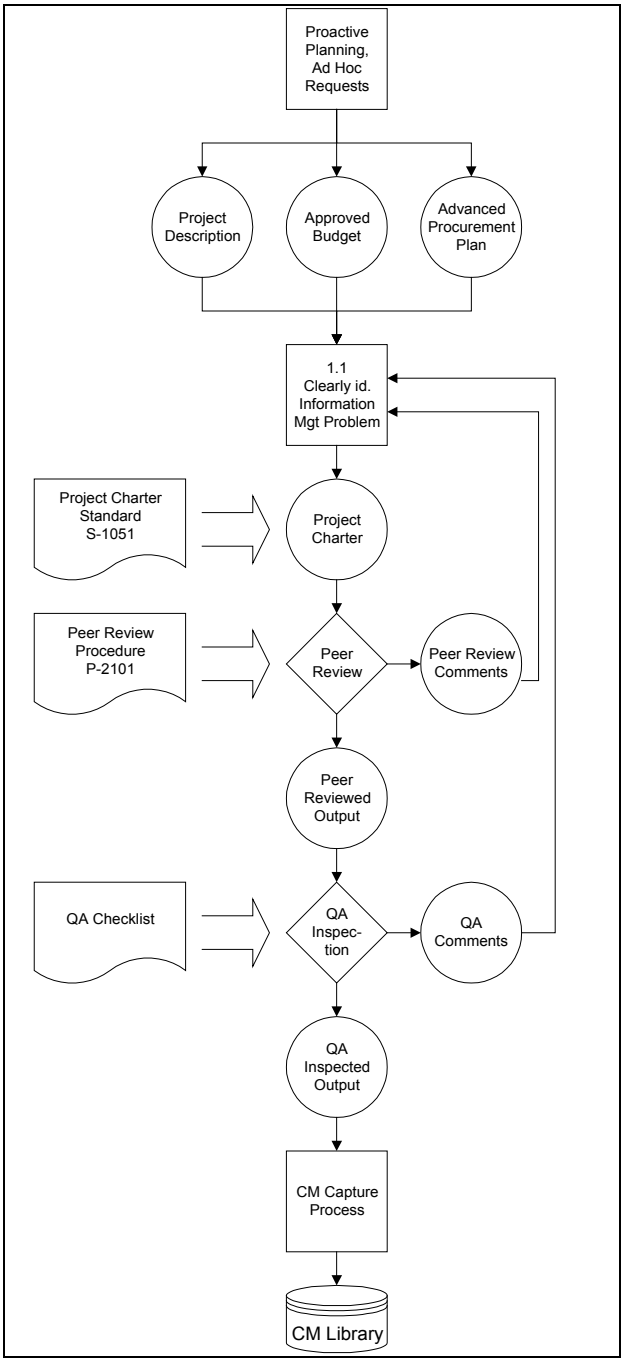


Figure 6–2. Identify Information Management Problem

Activity 1.2: Clarify Technical Scope

See Figure 6–3. Using the Project Charter developed in Activity 1.1, and the existing Technical Reference Manual and Enterprise Model,

1. Clarify the technical scope of the project.
2. Document this in the Scope section of the PDAD.

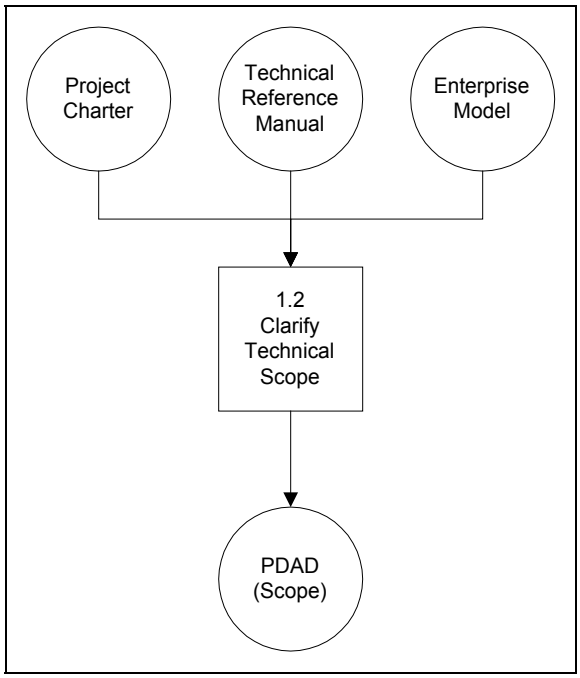


Figure 6–3. Clarify Technical Scope

Activity 1.3: Notify Records Management Branch to Review Records Management Requirements

See Figure 6–4. Using information from the Project Charter and the scope portion of the PDAD,

1. Complete the indicated government forms and deliver them to the Records Management Branch to notify them that an application system is being newly developed or enhanced.
2. After the forms have been processed by Records Management personnel, establish contact with the designated representative of the Records Management organization.
3. Add the legal and regulatory records management requirements to the user requirements to be documented in the PDAD in Activity 1.4

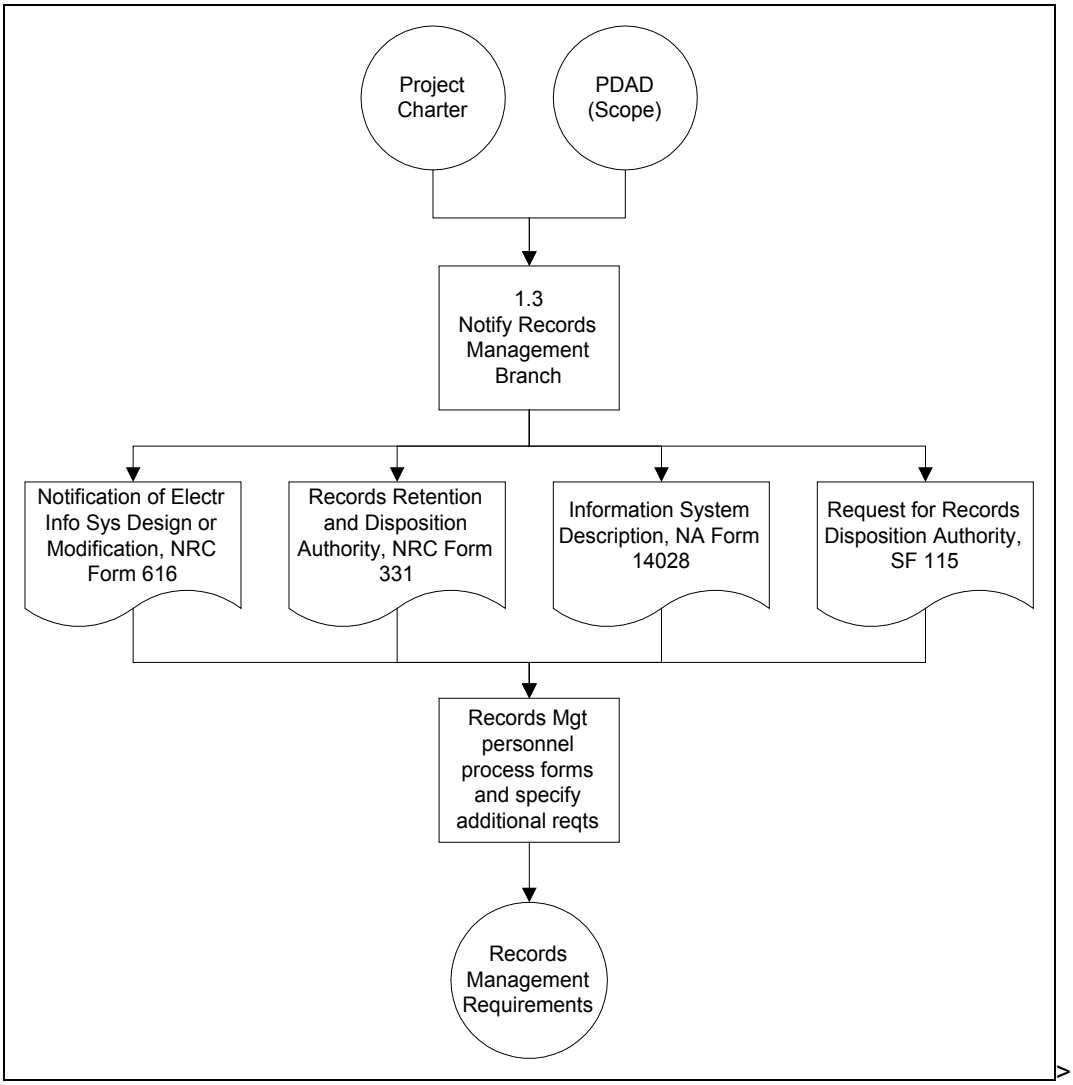


Figure 6–4. Notify Records Management

Activity 1.4: Identify Functional and Data Requirements

See Figure 6–5. Using the scope section of the PDAD, any additional input from Records Management, and the existing Technical Reference Manual and Enterprise Model,

1. Review the architecture in the Enterprise Model.
2. Document the relationships between the proposed project and the information system architecture in the Enterprise Model.
3. Review the current technology in the Technical Reference Model.
4. Assess the impact of the project on current technology architecture in the Technical Reference Model.
5. Develop a project context diagram to show how this project fits into the existing technical reference manual and enterprise model.
6. Develop an initial entity list.
7. Develop preliminary entity definitions for the entities listed above.
8. Evaluate legal requirements.
9. Evaluate regulatory requirements.
10. Evaluate public accessibility.
11. Identify physical requirements.
12. Identify data requirements.
13. Identify functional requirements.
14. Identify security requirements.
15. Identify human factors.
16. Package the results of your analysis in the Project Definition and Analysis Document, mapping the components of your analysis to the sections of the PDAD as defined in the standard.
17. Have QA review the PDAD to ensure that it meets requirements. Make corrections as needed.
18. Have CM capture the PDAD in a configured library.

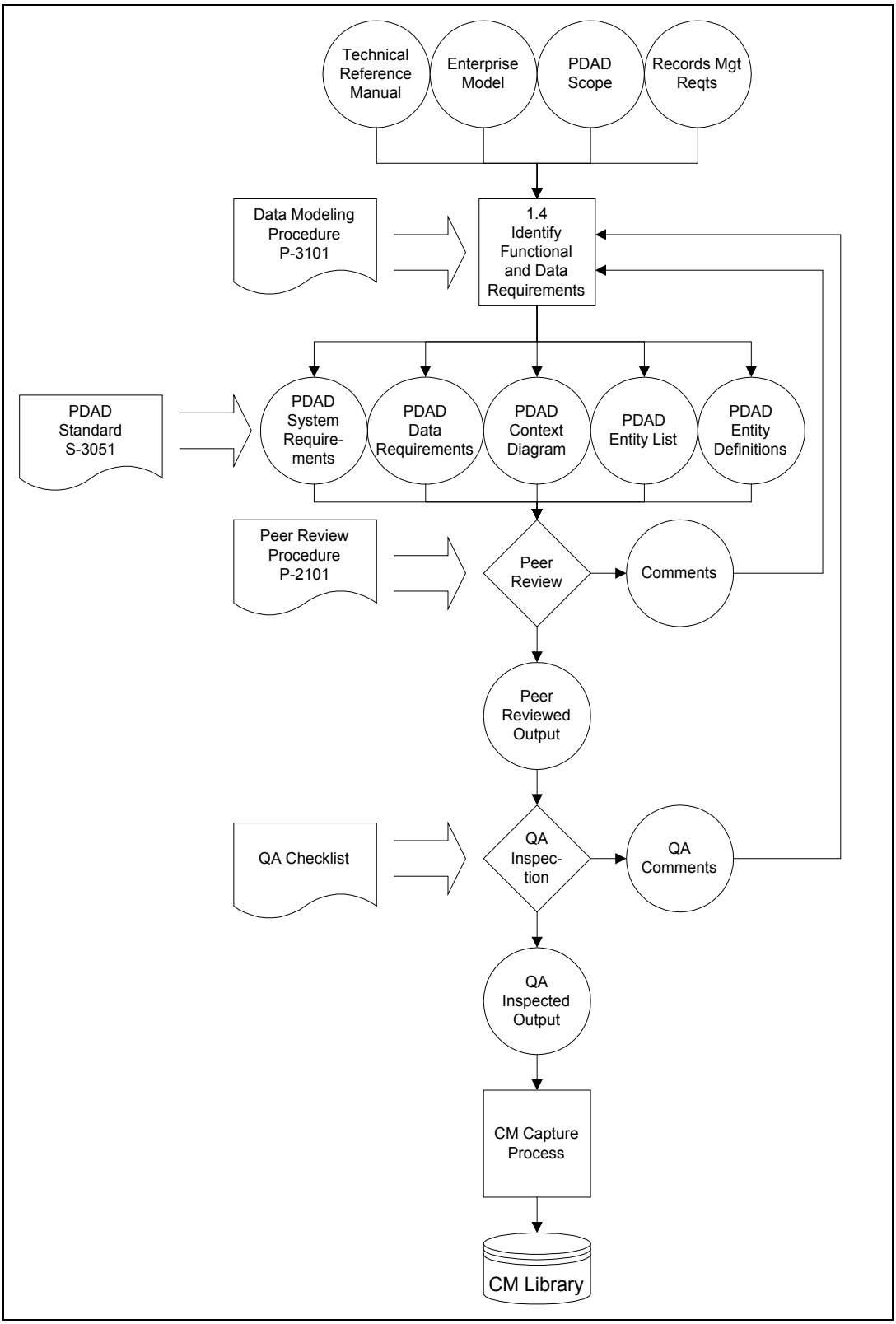


Figure 6–5. Identify Functional and Data Requirements

Activity 1.5: Analyze Alternatives

See Figure 6–6. This activity includes the assessment of current systems (if existing systems are being enhanced or replaced), the analysis of alternative approaches for satisfying the requirements, and the development of an operational concept.

Using the requirements documented in the PDAD, determine whether it is better to enhance, supplement, or replace any existing systems. Use a process such as the following:

1. Create a table for assessing the current system(s). Use one similar to the sample shown in the CSAD standard or in the "Assessment of Current System" section of the PDAD standard.
2. Review the list of functional requirements in the PDAD. Enter these functions in the Requirements column of the table.
3. Identify the existing systems that might be used to help solve the information management problem. For each existing system, determine which requirements are satisfied and enter those in the appropriate column of the table. Then note the remaining requirements satisfied by none of the current systems in the final column of the table.
4. Document the results of the assessment in the CSAD or directly in the PDAD. If there is no current system or if the assessment is relatively straightforward, the separate CSAD is not required.
5. Analyze the alternative approaching for satisfying the remaining requirements. Consider the advantages and disadvantages of enhancing, supplementing, or replacing each current system that was assessed. Document the results in the "Analysis of Alternatives" section of the PDAD.
6. Summarize the results of the analysis by developing a concept of operations of the proposed system. Document it in the "System Operations Concept" section of the PDAD.
7. Have QA review the Assessment of Current System, Analysis of Alternatives, and System Operations Concept; make any necessary corrections.
8. Have CM capture these documents in your configured library.

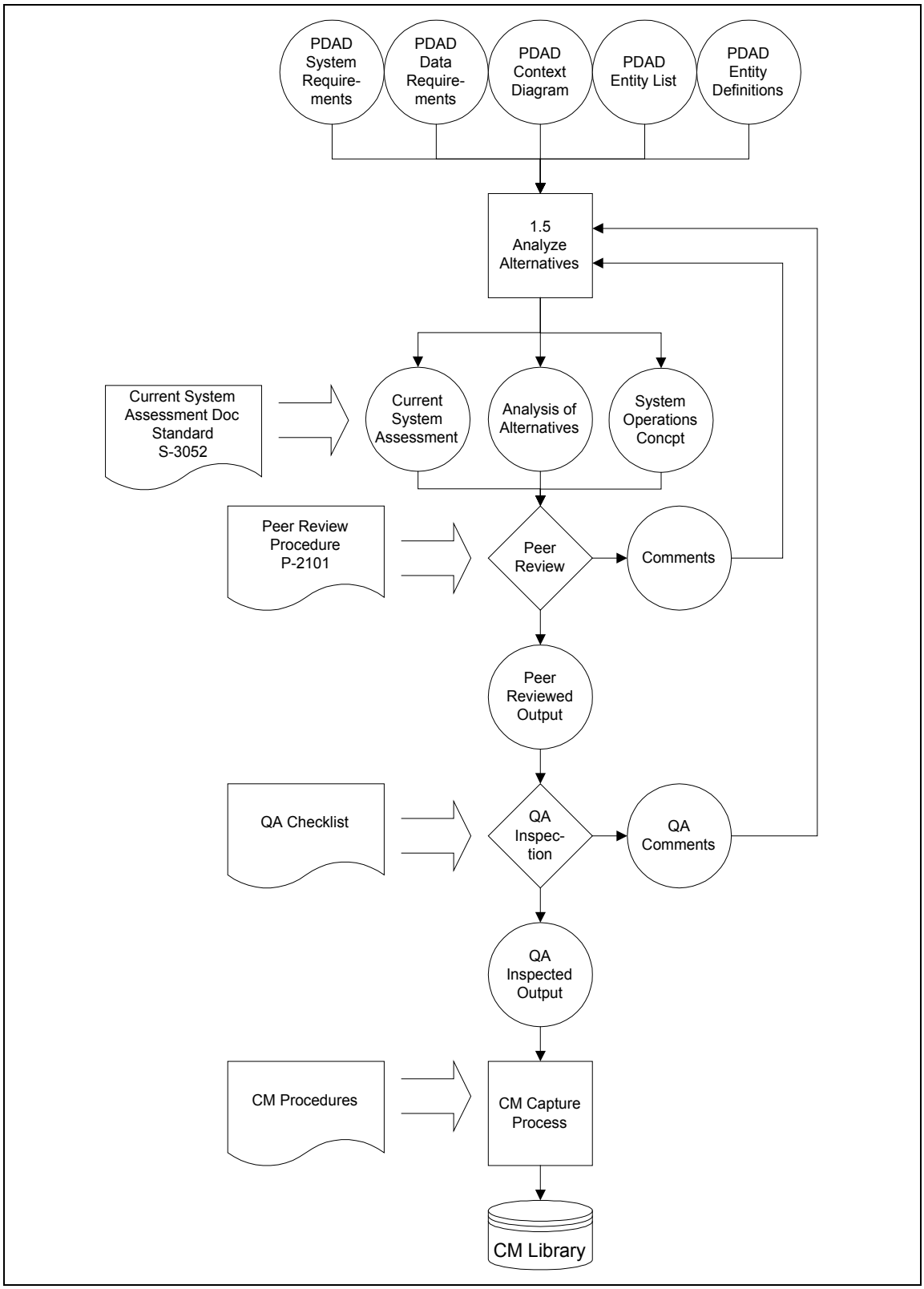


Figure 6–6. Analyze Alternatives

Activity 1.6: Establish a Project Plan

See Figure 6–7. Using the Project Charter, the PDAD, and the System Operations Concepts as input,

1. Develop a strategic plan. On the basis of the complexity of the project, choose an appropriate life-cycle model to develop the project. For example, if this is a very straightforward enhancement, a waterfall approach may be appropriate. If the requirements are not well defined, then an iterative development approach with active involvement of the customer may be more appropriate.
2. Use any tools and methods available to estimate the amount of effort required to develop the needed functionality.
3. On the basis of the amount of effort required and the complexity of the project, develop a tactical plan. That is, define a project team with the right mix of senior and junior people, and define their roles and responsibilities.
4. Define intermediate milestones and products.
5. Define project critical success measures, including performance requirements.
6. Evaluate and plan for records management.
7. Specify CM requirements.
8. Specify QA requirements.
9. Package the results of your work in a Project Action Plan.
10. Have QA review your plan and make any necessary changes.
11. Have CM capture your plan in the CM library.

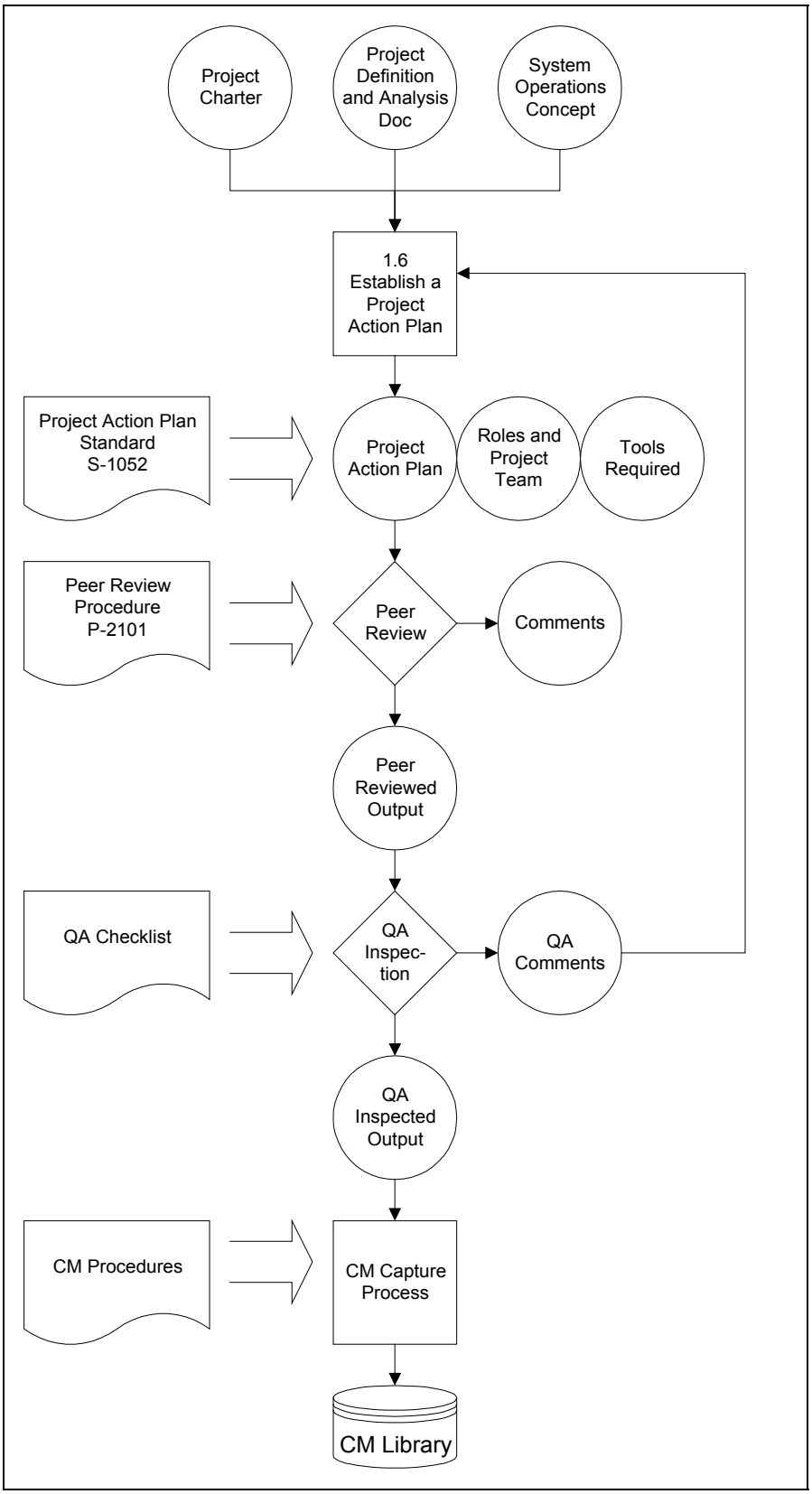


Figure 6–7. Establish a Project Plan

Activity 1.7: Review the Toolkit

See Figure 6–8. Using the results of your Analysis of Alternatives and the System Operations Concept:

- Review the Tool Inventory to select preferred tools or identify new tool requirements. This information will be used to develop your resources request below.

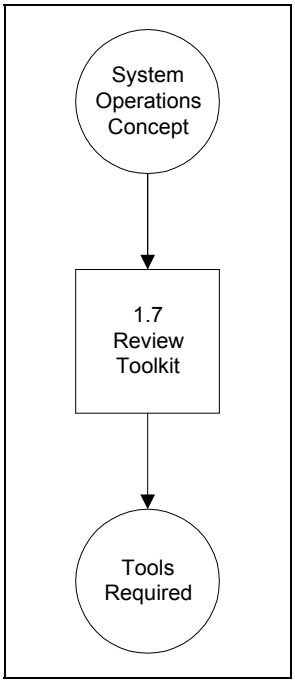


Figure 6–8. Review the Toolkit

Activity 1.8: Develop Support Resource Request

See Figure 6–9. Using the results of your detailed planning (as documented in the Project Action Plan), which shows the effort, people, computer hardware and software requirements, and any methods or tools necessary, request the resources you will need to be successful.

1. Use an Environment Change Request Form and the process defined in the Environment Change procedure to request any new tools or technologies.
2. Use a Support Resource Acquisition Request and Commitment Form, if needed, to request personnel and any other resources.
3. Identify procurement needs in detail.
4. Negotiate to obtain a commitment for the resources you need.

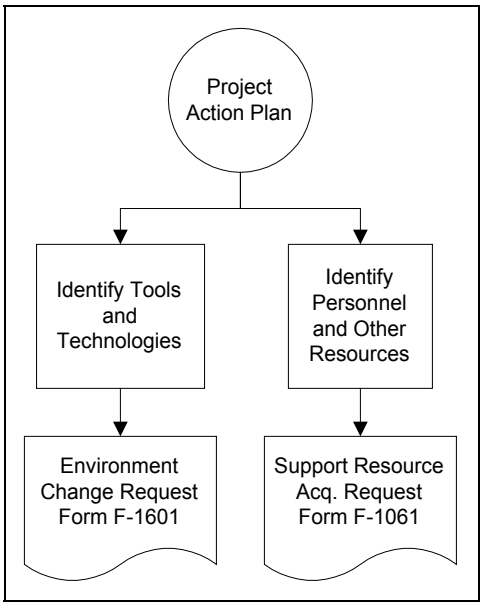


Figure 6–9. Develop Support Resource Request

Activity 1.9: Plan for Deployment

See Figure 6–10. Using all information provided by the user and all information gathered to support the development of the Project Charter, PDAD, and PAP, begin to plan for deployment by developing the initial version of the TIP. Document everything that is currently known about the deployment. Where necessary, indicate that sections of the TIP will be updated or completed as activities of subsequent components.

As suggested by Figure 6–1 (see Page 31), development of the TIP may be started concurrently with development of the other major products of Component 1

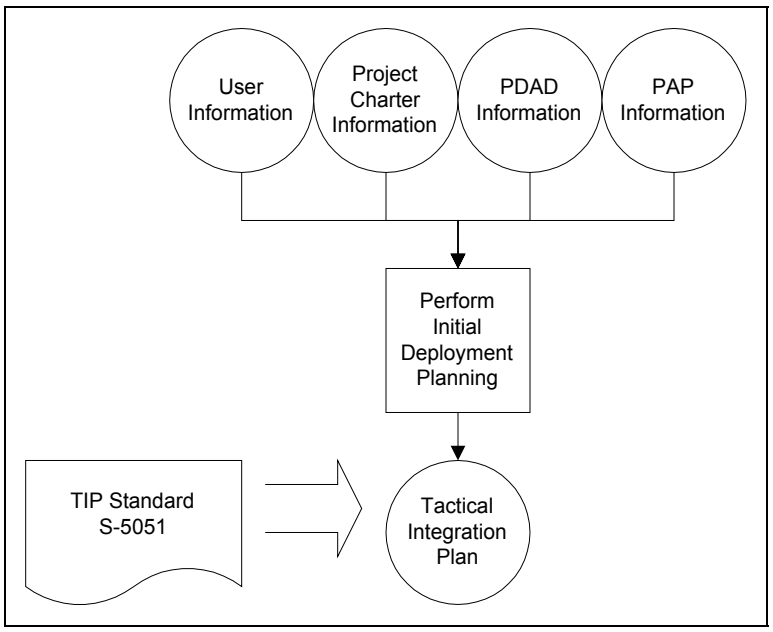


Figure 6–10. Plan for Deployment

Activity 1.10: Review (Component 1)

See Figure 6–11. Review the project to make a GO or NO-GO decision.

1. Justify the project. Review the need for the project, the cost in terms of personnel and resources, and the return on investment for the project. Consider the best case scenario and the worst case scenario of moving forward with the project.
2. Complete the form required obtaining approval to proceed.

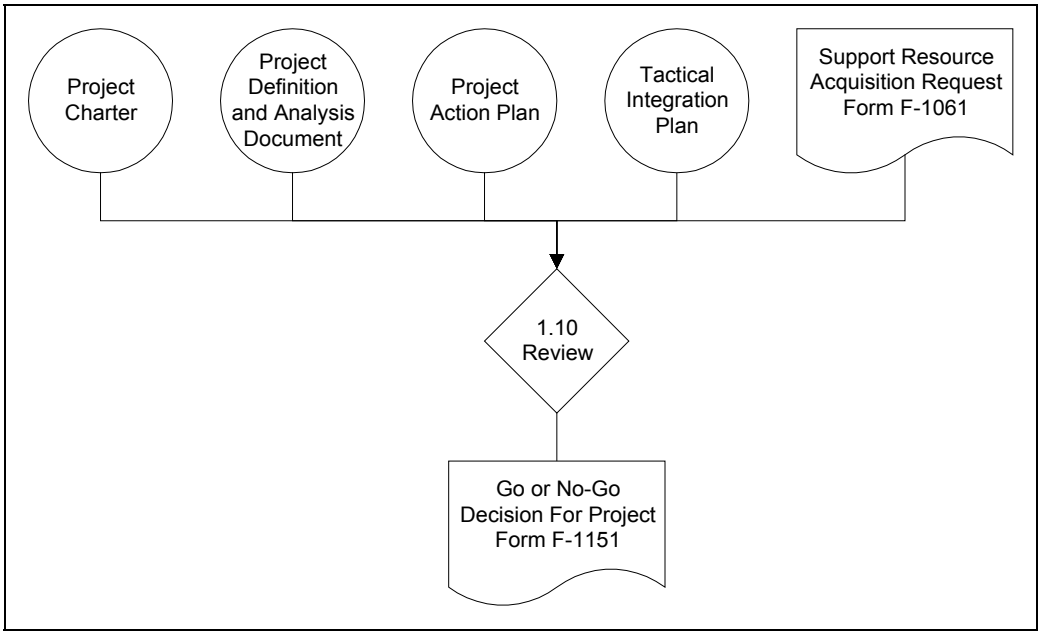


Figure 6–11. Review Component 1

7. Component 2. Acquire Support Resources

7.1 Purpose

The purpose of this component is to obtain the necessary resources to ensure timely and effective progress on the project, including:

- Staff
- Technology
- Contractors
- Training

7.1.1 Projects and Funding Vehicles

It is very important to understand that NRC's SDLCM Methodology deals with projects and products, *not* with contracts and deliverables. This section of the chapter on Component 2, Acquire Support Resources, relates the project-oriented methodology to the contracting process. The remainder of the documentation set of the SDLCM Methodology is free of contract-dependent terminology, so it can be applied on any contract vehicle.

NRC's typical mechanism for providing a source of project funding to a contractor is the Task Assignment Control (TAC).

The three terms *TAC*, *project*, and *system* have three distinct meanings:

- A *TAC* a vehicle for funding all of a project, part of a project, or multiple projects.
- A *project* is an undertaking requiring an organized effort, focused on developing or maintaining a specific product. Typically, a project has its own funding, cost accounting, and product schedule. That is, the project is the work to be done and the personnel assigned to perform the work; it is not the same as the product (for example, a *system*) that the project personnel are to produce.
- A *system* is an operational entity. An application system or information system supports a business requirement.

A project produces *products*. A TAC requires the delivery of *deliverables*.

NRC designed the SDLCM Methodology to be applied to a project rather than to a TAC.

7.1.2 Funding Approaches and Project Products

A project may be funded all at once under a single TAC, incrementally under a single TAC, or incrementally under a sequence of TACs.

The document products specified by the SDLCM Methodology are to be developed as products of a complete project, not necessarily as deliverables of a single TAC. If a TAC funds only a portion of a project, it then stands to reason that not all project products will be delivered under that TAC. In any case, the project manager is responsible for ensuring that all project products (as specified by the SDLCM Methodology) are produced (or formally waived) whether or not they are contractually specified as TAC deliverables. The Project Action Plan, for example, is a

key project management tool, which must be updated whenever any change results from the issuance of a new TAC or the modification to an existing TAC.

The ideal contracting approach (see Figure 7–1) is to fund a single project under a single TAC (perhaps with incremental additions of funds). For example, a possible approach is to fund Component 1, so that the contractor can help with the definition of the requirements, and then to fund and execute Component 2 to modify the funding vehicle (not to issue a different one) to acquire support for Components 3, 4 and 5 as a unit. Separating 3 and 4 and funding them separately is not reasonable because together they comprise the software development life cycle portion of the SDLCM Methodology (see Chapter 3). Depending on the life-cycle model chosen (again see Chapter 3 for examples of life-cycle models), it may be logically impossible to separate Components 3 and 4. If the same contractor supports the system deployment in Component 5, or if the deployment is phased along with the design and engineering aspects, then it is also not feasible to separate Component 5 and all three components must be treated as a unit under a single TAC.

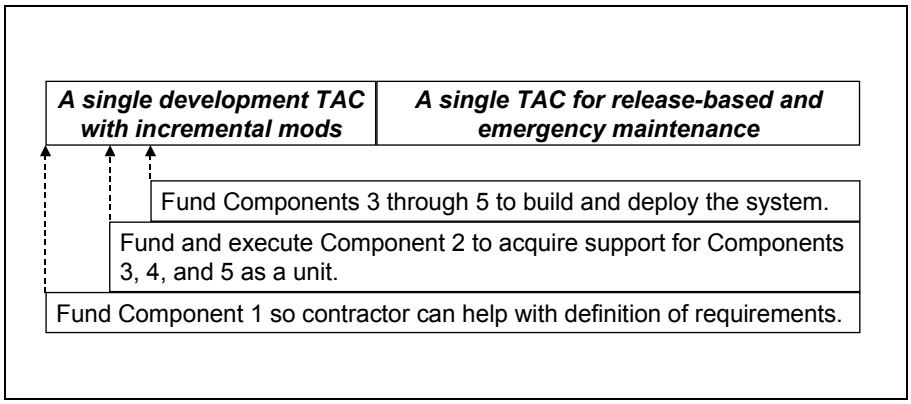


Figure 7–1. Ideal Project Funding Approach

7.2 Roles and Responsibilities

The roles required to perform the activities in the Acquire Support Resources component are shown in Entry Criteria

The following of the following may trigger the initiation of Component 2:

- Approved project from the Define Initial Project Requirements component
- Change of priorities
- Change in commitments of staff

Table 7–1.

7.3 Entry Criteria

The following of the following may trigger the initiation of Component 2:

- Approved project from the Define Initial Project Requirements component
- Change of priorities
- Change in commitments of staff

Table 7–1. Component 2 Roles and Responsibilities

Roles	Responsibilities
Overall Project Manager Business Project Manager Technical Project Manager	Accountable for Execution
Executive Sponsor	Approval to Proceed
Business Advocate Development Team Business SME Technical SME Other Representatives or Stakeholders • Quality Assurance Manager • Configuration Management Manager	Provides Input

7.4 Input, Activities, and Outputs

The inputs to this component are as follows:

- Project Action Plan
- Project Definition and Analysis Document
- Tactical Integration Plan
- Support Resource Acquisition Request and Commitment
- New tool requirement
- Staffing (current and planned)
- Advance Procurement Plan (outside the scope of this methodology)
- Budgets (outside the scope of this methodology)

Figure 7–2 illustrates the flow of the following activities:

1. Specify the work to be done
2. Staff the project
3. Train the staff
4. Acquire and install other required resources
5. Update the Project Action Plan
6. Continue Deployment Planning
7. Review (Component 2)

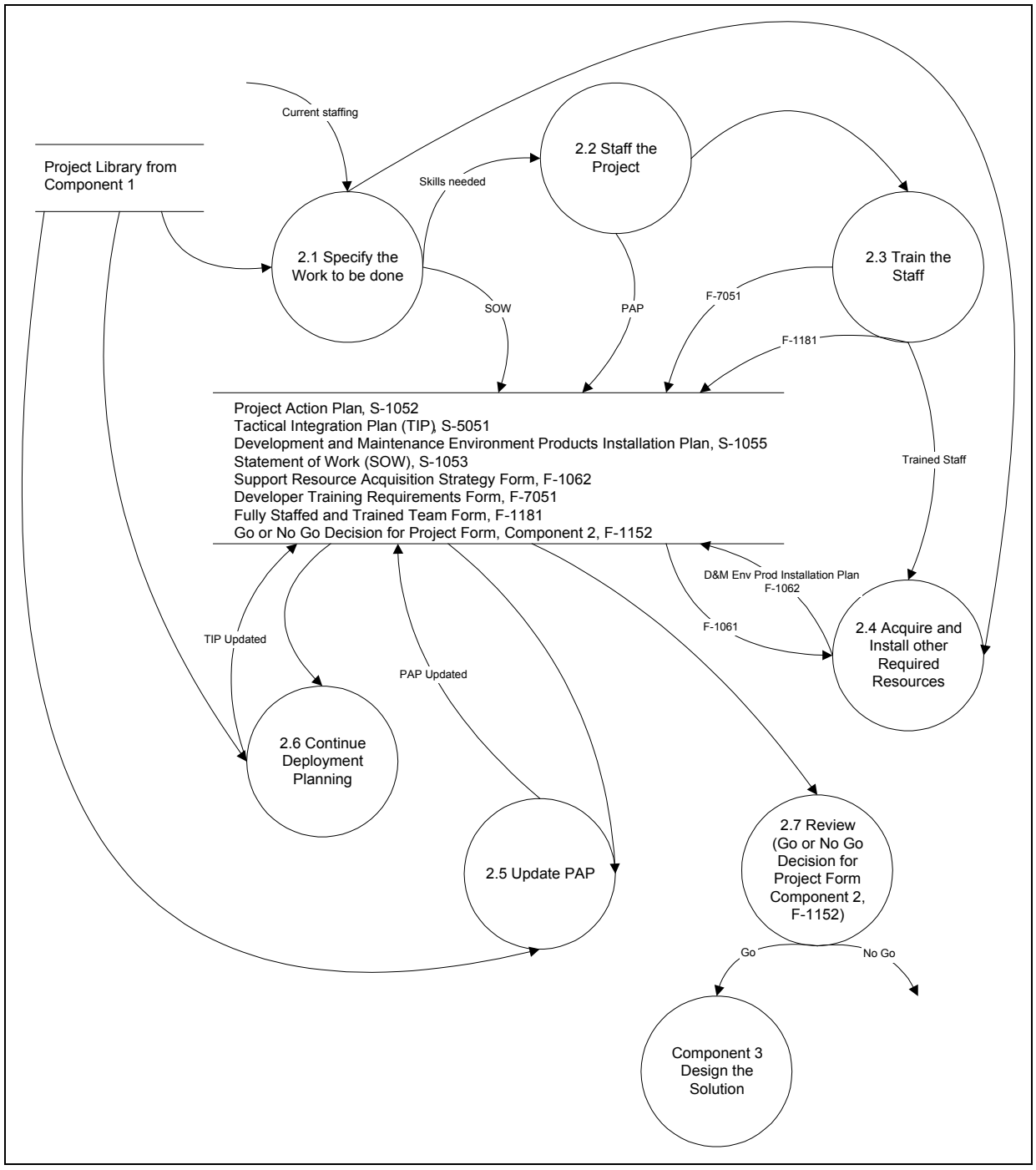


Figure 7–2. Component 2

The inputs to this component were produced as products of Component 1 or prior to the start of the project and are in the project library.

As the figure suggests, many of the activities can be performed in parallel. The activities are described in more detail in Section 7.7.

Figure 7–2 also summarizes the outputs of all Component 2 activities in the central data store.

Appendix C includes a summary description of all products of NRC projects, indicates within which components each product is created and updated, and specifies which products are required.

The products, whose standard and form numbers are shown in the figure, are described in more detail in the companion volume *SDLCM Methodology Procedures, Standards, and Forms*.

7.5 Techniques and Tools

The activities of Component 2 are supported by the following techniques.

- Structured Walkthrough
- Peer Review

Refer to the current *SDLCM Methodology Tool Inventory* for the recommended set of tools.

7.6 Exit Criteria

Component 2 is complete when:

- All critical products have been reviewed (structured walkthrough or peer review)
- Key products have been inspected by QA to conform to project standards
- Products have been placed under CM control
- Information has been shared
- Issues have been resolved
- There is management buy-in to move forward
- There are no reasons to stop this project now

7.7 Component 2 Activity Details

The following pages provide detailed activities of Component 2, Acquire Support Resources.

Activity 2.1: Specify the Work to be Done

See Figure 7–3. Using the products of Component 1 as input, prepare a statement of work and submit it to a contractor using a contractual funding vehicle. (As explained in Section 7.1, discussion of contracts and funding vehicles is outside the scope of the SDLCM Methodology.)

1. Create a work breakdown structure that specifies the work to be done by both NRC staff and contractors. Develop a matrix showing what work needs to be done and by whom. See Table 7–2.
2. Using the work breakdown structure, create a statement of work for contracted resources.

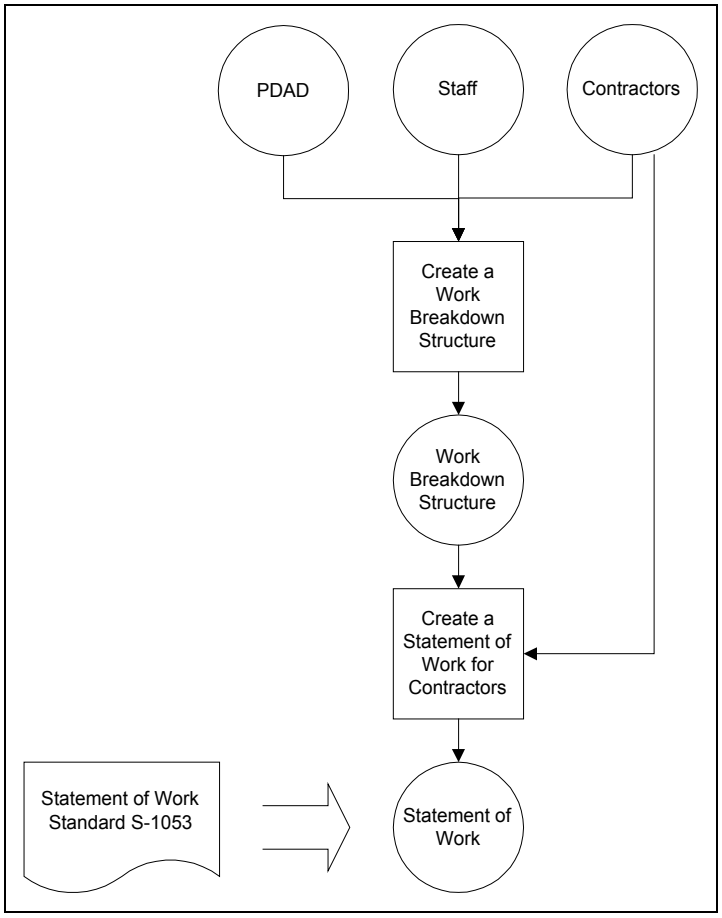


Figure 7–3. Specify the Work to be Done

Table 7–2. Work Matrix

NRC Staff	Contractors	Work to be Done
x		Task 1
	x	Task 2
x		Task *n*

Activity 2.2: Staff the Project

See Figure 7–4. Using the PDAD (for an understanding of the technical skills of the people required), an awareness of the current staff, planned staff, and budget:

1. Determine any additional staffing requirements needed for the project to be a success.
2. Acquire additional staff, using contractors as needed and budgeted for
3. Establish team rules of operation and document them in the Project Action Plan

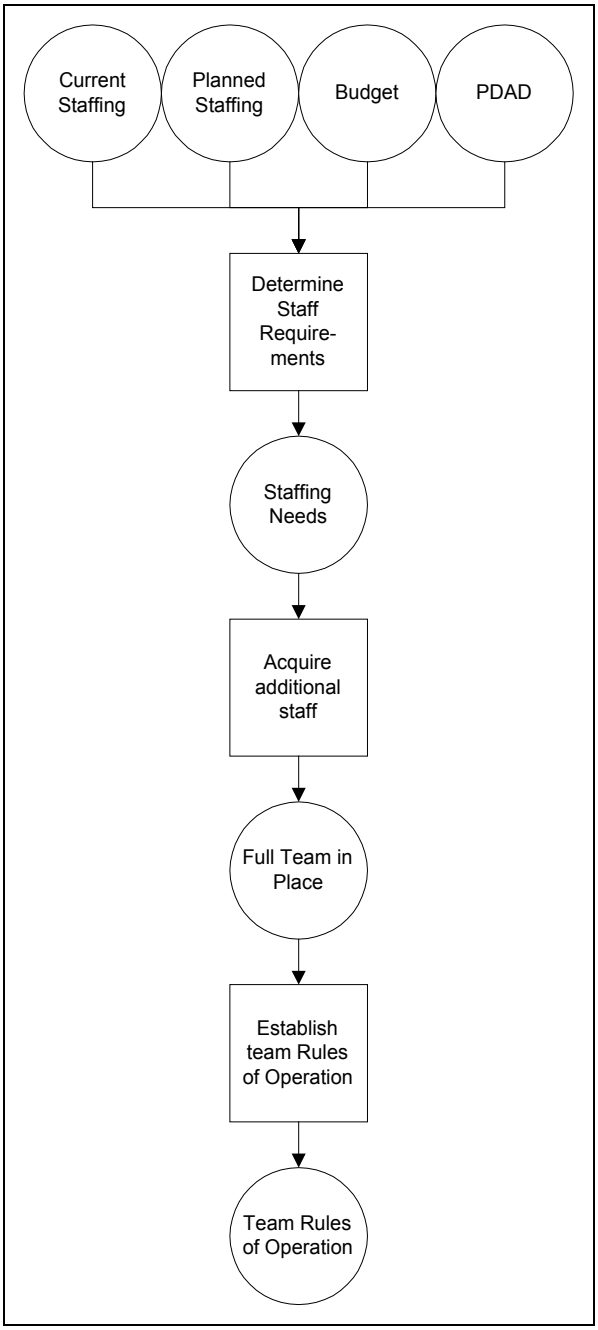


Figure 7–4. Staff the Project

Activity 2.3: Train the Staff

See Figure 7–5. Given a staff with existing skills, and a knowledge of the kinds of skills needed (based on an analysis of the PDAD):

1. Determine what training is required
2. Procure the training
3. Perform the training

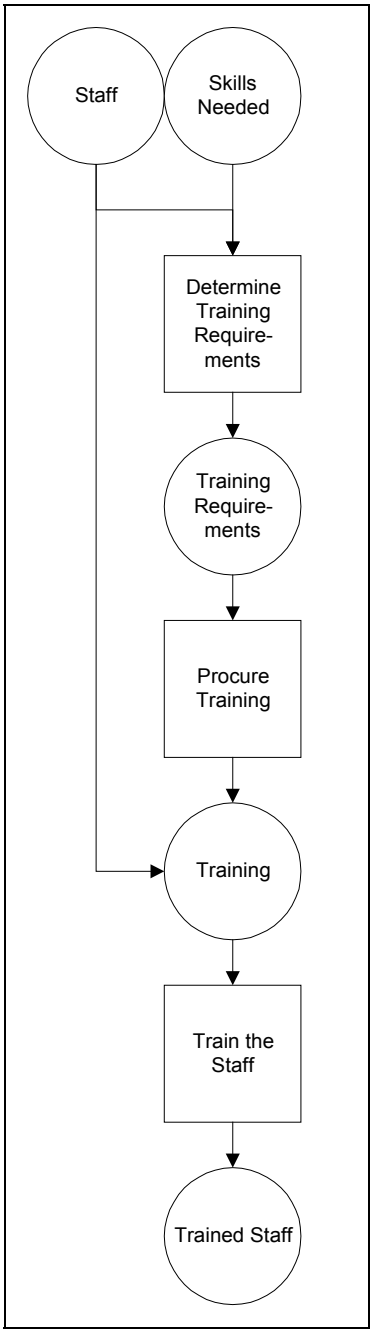


Figure 7–5. Train the Staff

Activity 2.4: Acquire and Install Other Required Resources

See Figure 7–6. Using the Support Resources Acquisition Commitments obtained from Component 1, as well as the Advanced Procurement Plan, budget information, and Tool Request forms:

1. Select an acquisition strategy for additional resources
2. Obtain items within budget
3. Prepare a plan to install resources needed for the development and maintenance environment
4. Install the support resources

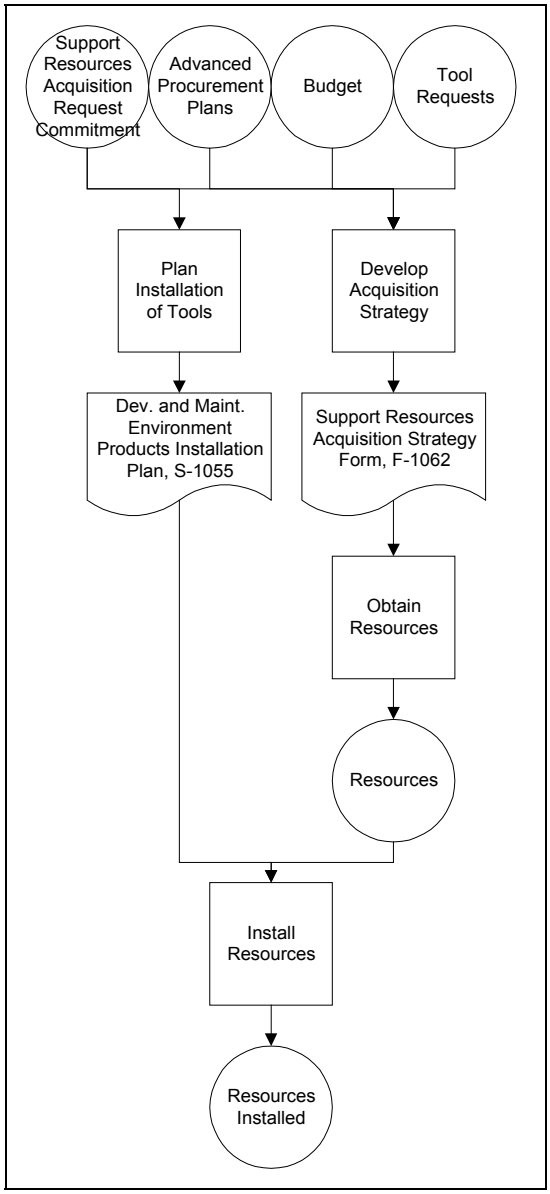


Figure 7–6. Acquire and Install Other Required Resources

Activity 2.5: Update the Project Action Plan

See Figure 7–7. Using the current status of the project staff and other resources,

1. Review the Project Action Plan
2. Update the Project Action Plan so that schedules, staffing profiles, risk assessments, and any other management planning matters reflect the current status

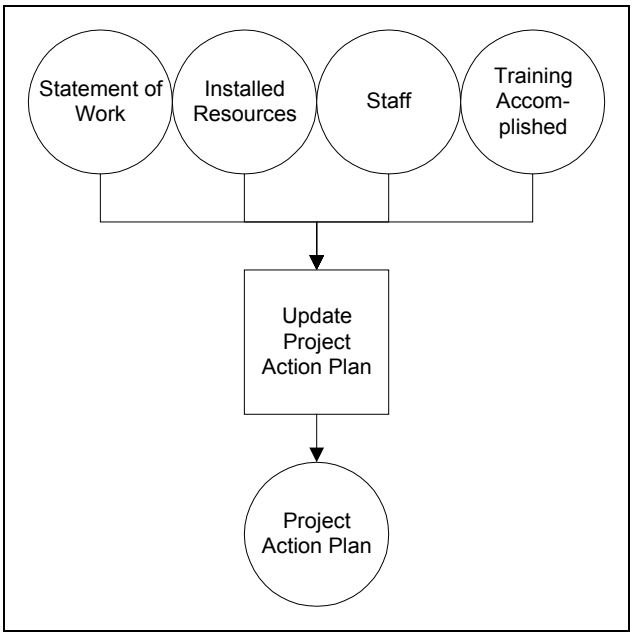


Figure 7–7. Update the Project Action Plan

Activity 2.6: Continue Deployment Planning

See Figure 7–8. Using the current status of the project,

1. Review the Tactical Integration Plan
2. Update the Tactical Integration Plan so that deployment schedules and any other deployment matters reflect the current status

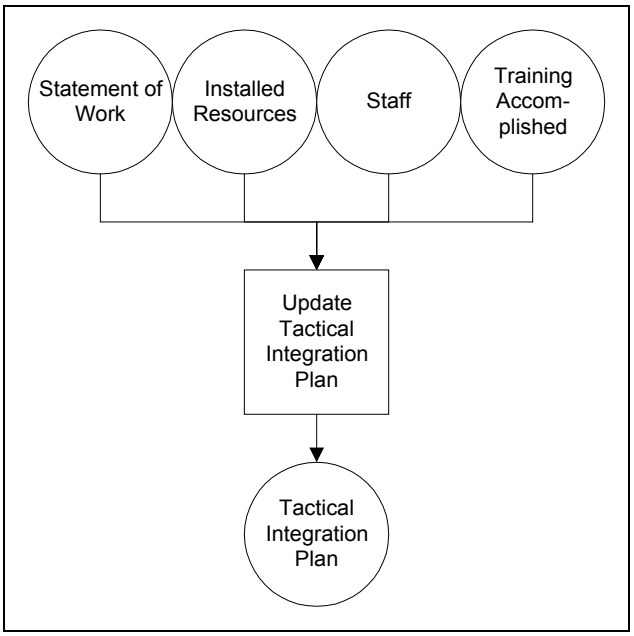


Figure 7–8. Continue Deployment Planning

Activity 2.7: Review (Component 2)

See Figure 7–9. Review the project to make a GO or NO-GO decision.

1. Review for the need for the project and the project status
   - Has the work been specified?
   - Are all equipment resources operational before we move forward with the project?
   - Is the project team in place?
   - Is the team trained and ready to begin work?
   - Is there sufficient organizational commitment and executive sponsorship to move forward with the project?
2. Complete the form required to obtain approval to proceed.

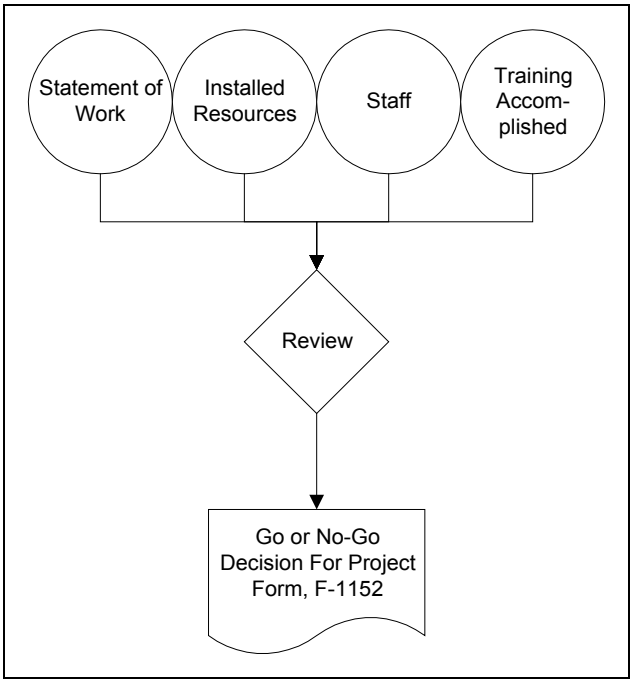


Figure 7–9. Review Component 2

8. Component 3. Design the Solution

8.1 Purpose

The purpose of this component is to:

- Determine functional requirements
- Determine data requirements
- Determine communication requirements
- Determine network requirements
- Document user interface requirements
- Design the solution
- Prepare integration plans
- Prepare project plans
- Prepare training plans
- Secure a Go Decision to proceed with engineering the solution

8.2 Roles and Responsibilities

The roles required to perform the activities in the Design the Solution component are shown in Table 8–1.

Table 8–1. Component 3 Roles and Responsibilities

Roles	Responsibilities
Overall Project Manager Business Project Manager Technical Project Manager Development Team	Accountable for Execution
Executive Sponsor Business Advocate	Approval to Proceed
Business Advocate Business SME Technical SME Other Representatives or Stakeholders • Quality Assurance Manager • Configuration Management Manager	Provides Input

8.3 Entry Criteria

The following events may trigger the initiation of Component 3:

- Completion of Component 2, Acquire Support Resources
- Receipt of additional requirements

8.4 Input, Activities, and Outputs

The inputs to this component are as follows:

- Project Action Plan
- Project Definition and Analysis document
- Tactical Integration Plan
- Fully staffed and trained project team
- Installed resources for design
- Infrastructure
- Selected technology and tools
- Budget allocation
- Enterprise Model
- Technical Reference Model

Figure 8–1 illustrates the flow of the following activities:

1. Analyze Requirements and Perform High Level Design
2. Plan Solution Integration
3. Design Training Materials
4. Establish Test Approach
5. Update Project Action Plan
6. Continue Deployment Planning
7. Review (Component 3)

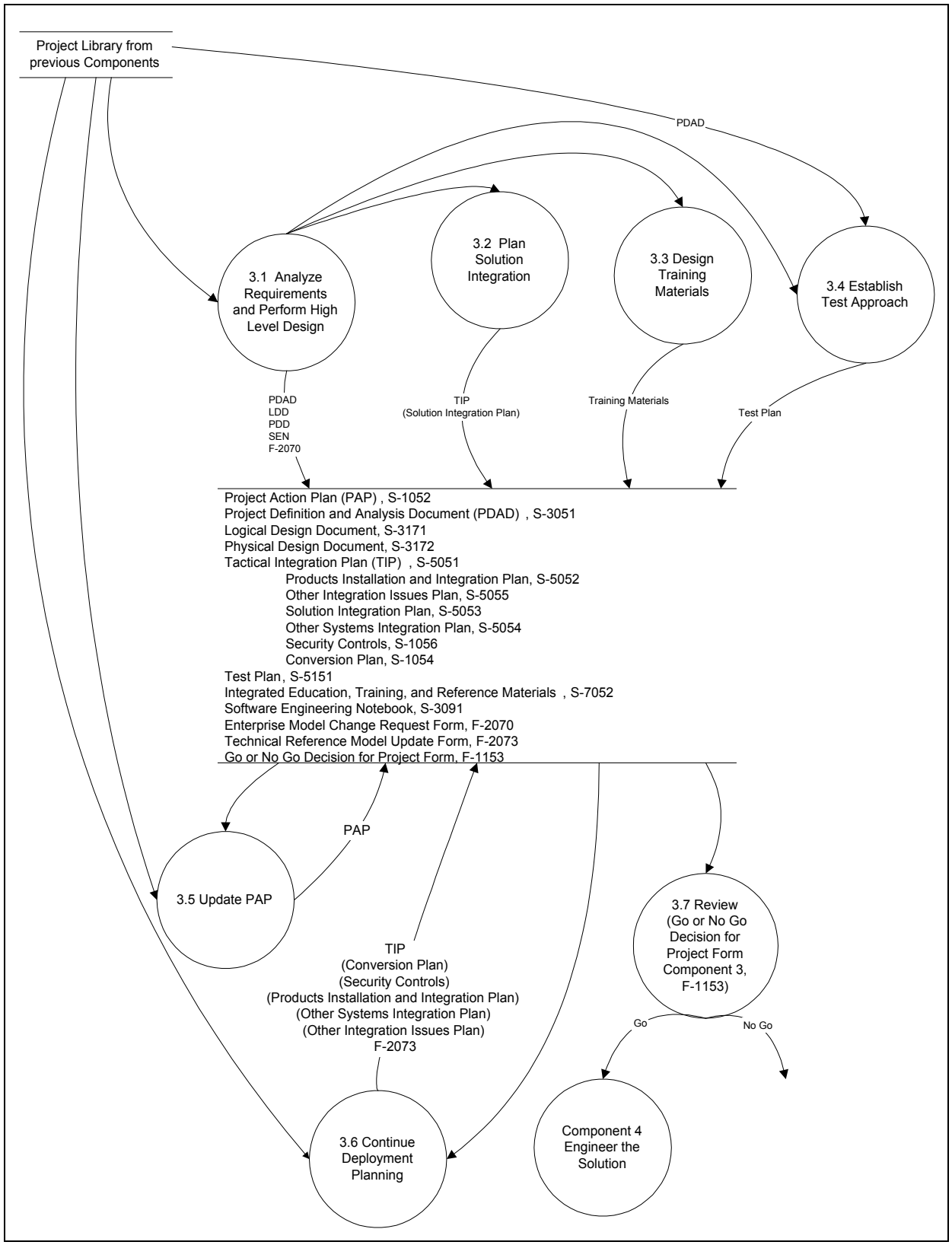


Figure 8–1. Component 3

The inputs to this component were produced as products of earlier components.

As the figure suggests, many of the activities can be performed in parallel. The activities are described in more detail in Section 8.7.

Figure 8–1 also summarizes the outputs of all Component 3 activities in the central data store.

Appendix C includes a summary description of all products of NRC projects, indicates within which components each product is created and updated, and specifies which products are required.

The products, whose standard and form numbers are shown in the figure, are described in more detail in the companion volume *SDLCM Methodology Procedures, Standards, and Forms*.

8.5 Techniques and Tools

The activities of Component 3 are supported by the following techniques.

- Data Modeling
- Joint Application Design and Development
- Process Modeling
- Prototyping
- Structured Facilitation
- Structured Walkthrough
- Peer Review

Refer to the current *SDLCM Methodology Tool Inventory* for the recommended set of tools.

8.6 Exit Criteria

Component 3 is complete when:

- All critical products have been reviewed (structured walkthrough or peer review).
- Key products have been inspected by QA to conform to project standards.
- Products have been placed under CM control.
- Information has been shared.
- Issues have been resolved.
- There is management buy-in to move forward.
- There are no reasons to stop this project now.

8.7 Component 3 Activity Details

The following pages provide detailed activities of Component 3, Design the Solution.

Activity 3.1: Analyze Requirements and Perform High Level Design

This activity is divided into six sub-activities described on the following pages:

3.1.1 Analyze Network

3.1.2 Analyze Data

3.1.3 Analyze and Design Processes

3.1.4 Analyze and Design User Interface

3.1.5 Analyze Platform and Operation System

3.1.6 Design for Functional Requirements

Activity 3.1.1: Analyze Network

See Figure 8–2. Using the Project Charter and the Project Definition and Analysis Document (which includes the System Operations Concepts), determine

- What locations must be supported
- What organizations must be supported
- What are the synchronization requirements

Use this information as input to the subsequent process analysis sub-activity.

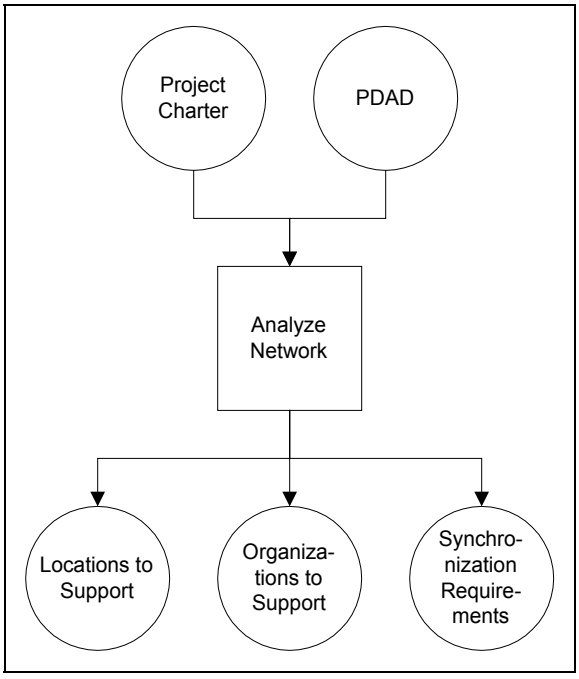


Figure 8–2. Analyze Network

Activity 3.1.2: Analyze Data

See Figure 8–3. Using the Technical Reference Model, the Enterprise Model, and the current version of the Project Definition and Analysis Document (which includes the entity list and definitions, the context diagrams, and the System Operations Concept), perform a more detailed data analysis and follows:

1. Identify all data requirements:
   - Identify corporate data that can be reused verbatim
   - Identify corporate data that can be reused once converted
   - Identify data that must be added
2. From this analysis, create the logical design
   - Specify entity types
   - Specify attribute types
   - Specify entity-relationships
3. Once all entities, attributes, and relationships are established, normalize the database to eliminate redundancy and optimize maintainability.
4. Create the physical design:
   - Map entity types to relational tables
   - Map attribute types to columns
   - Map relationship types to relational schema relationships
5. Calculate the entity volumes.
6. From the entity volumes, compute the data capacity and archiving requirements.
7. Establish security requirements.
8. Select design validation approach.
9. Validate the design.
10. Update the Enterprise Model.

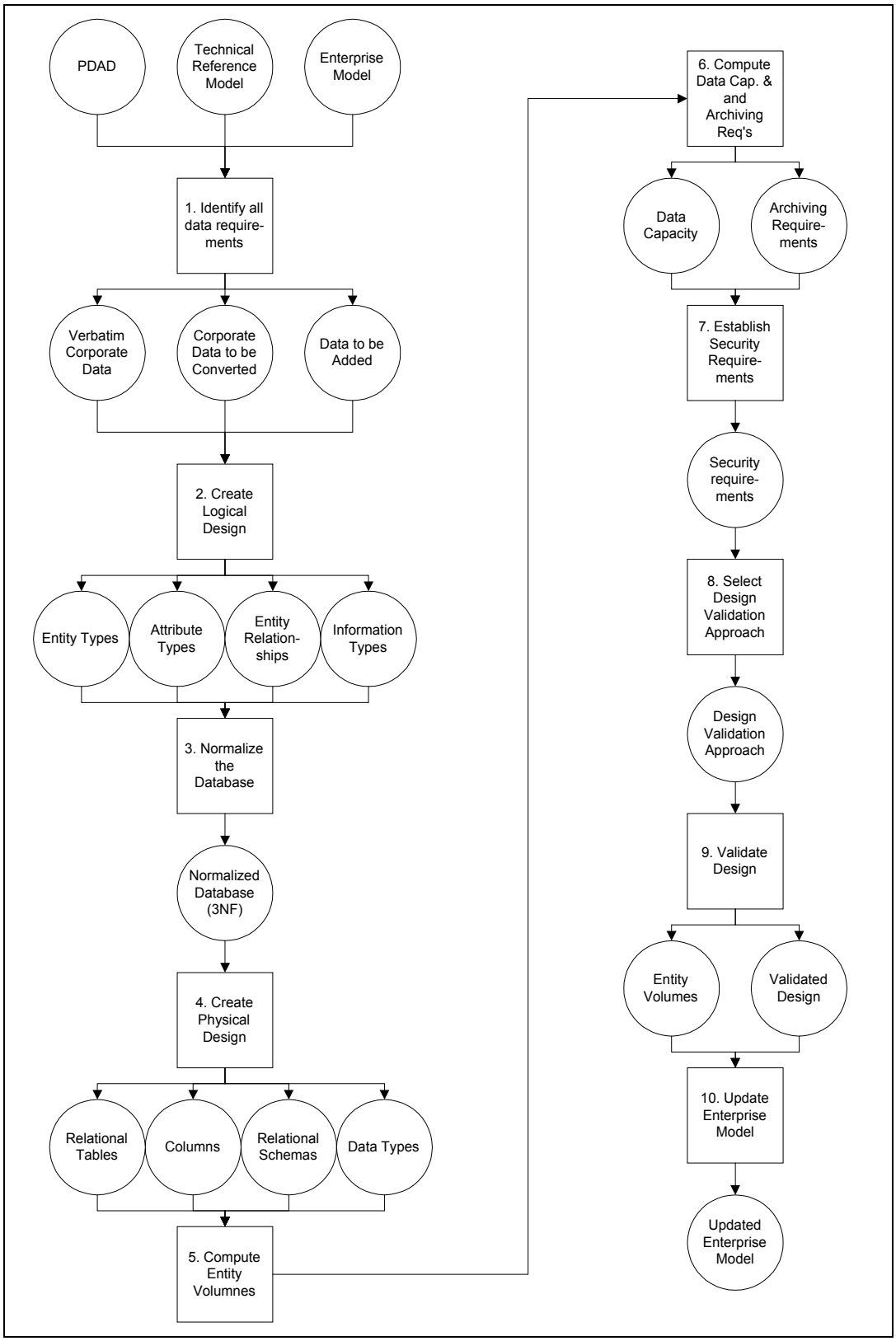


Figure 8–3. Analyze Data

Activity 3.1.3: Analyze and Design Processes

See Figure 8–4. Using the System Operations Concept from the PDAD and the results of the previous Data Analysis sub-activity, analyze the processes required for the information management system and identify what processes should be automated and which ones should be manual. Document the results of the analysis and design in the logical and physical design documents. Specifically:

1. Identify business procedures, practices, and decision-making activities that must be a part of the information management system. Include the records management requirements of the business.
2. Identify security requirements.
3. Identify external events.
4. Merge the business procedures, security requirements, and external events with the software and hardware part of the information management system to create a data flow diagram.
5. Identify complex processes that must be decomposed and modeled.
6. Decompose and model complex processes.
7. From the processes, determine which processes should be manual and which should be automated.
8. Establish performance requirements by type of transaction (volume, speed, frequency, etc.)

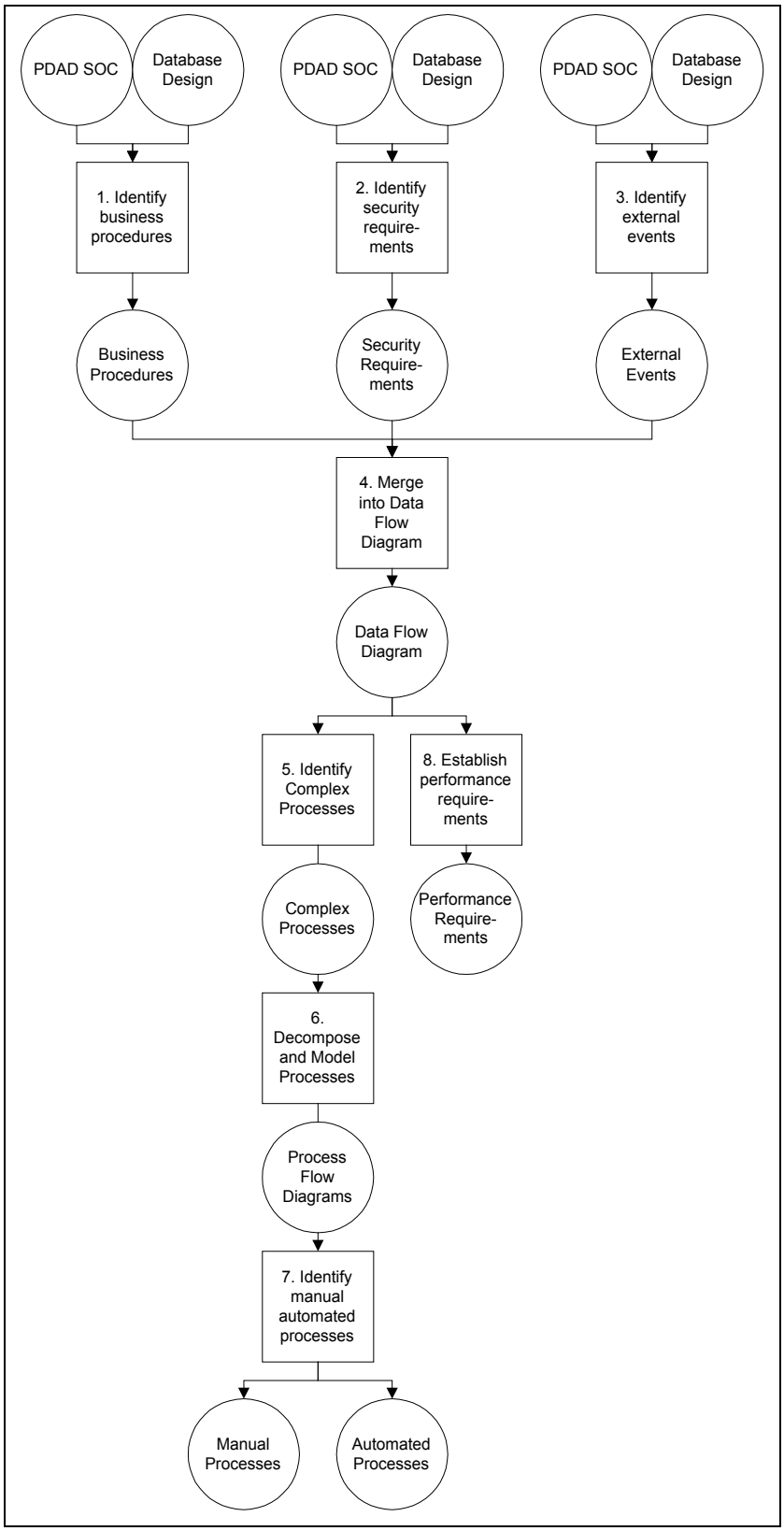


Figure 8–4. Analyze and Design Processes

Activity 3.1.4: Analyze and Design User Interface

See Figure 8–5. Using the results of the database design and process design from previous sub-activities, design the user interface as follows:

1. Determine data entry requirements
2. Determine display requirements
3. Determine dialog requirements
4. Determine screen requirements
5. Create screen prototypes
6. Establish performance requirements

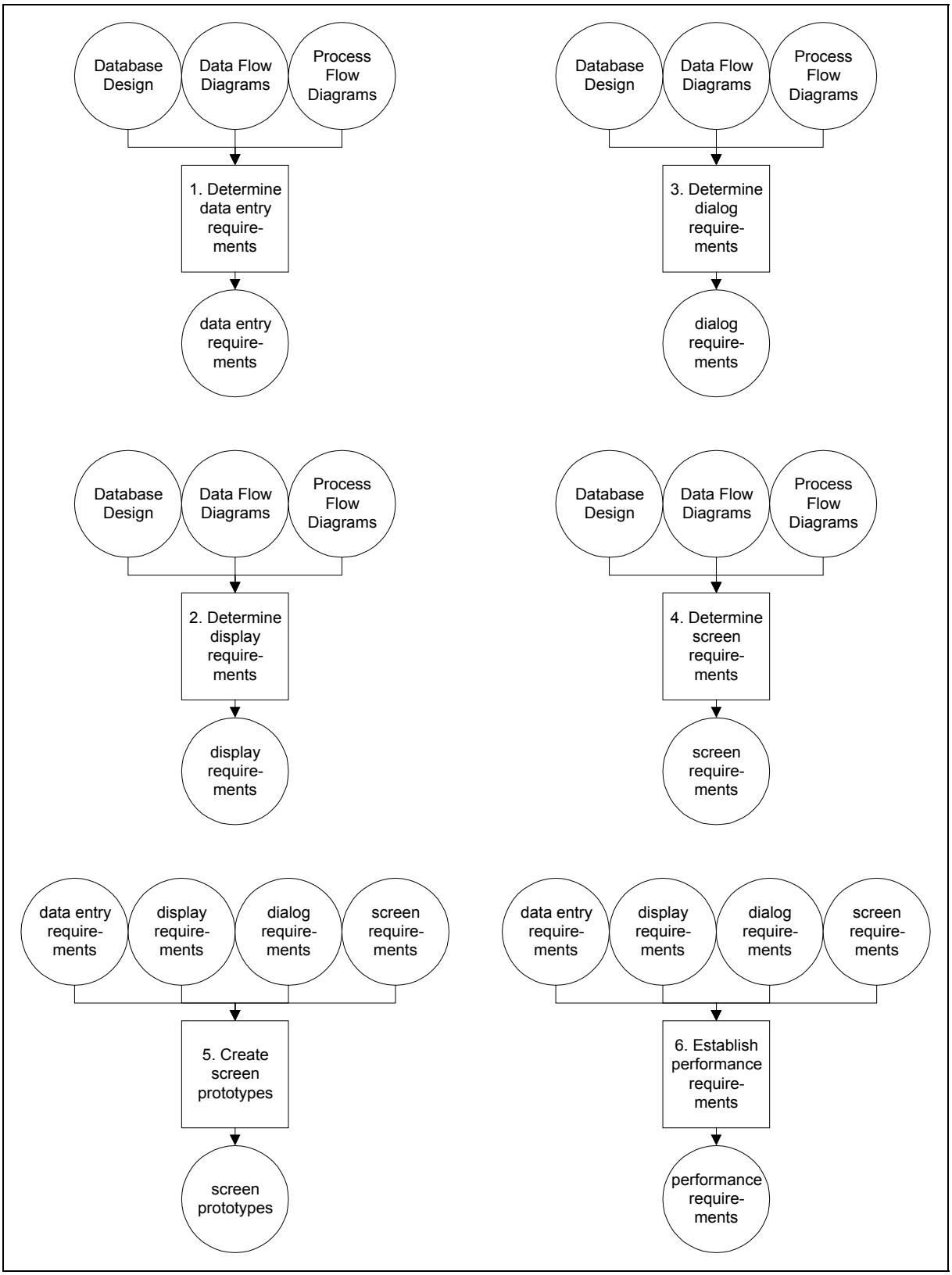


Figure 8–5. Analyze and Design User Interface

Activity 3.1.5: Analyze Platform and Operation System

See Figure 8–6. Using the results of the data analysis, process analysis, and user interface analysis, determine the hardware and operating system requirements of the system.

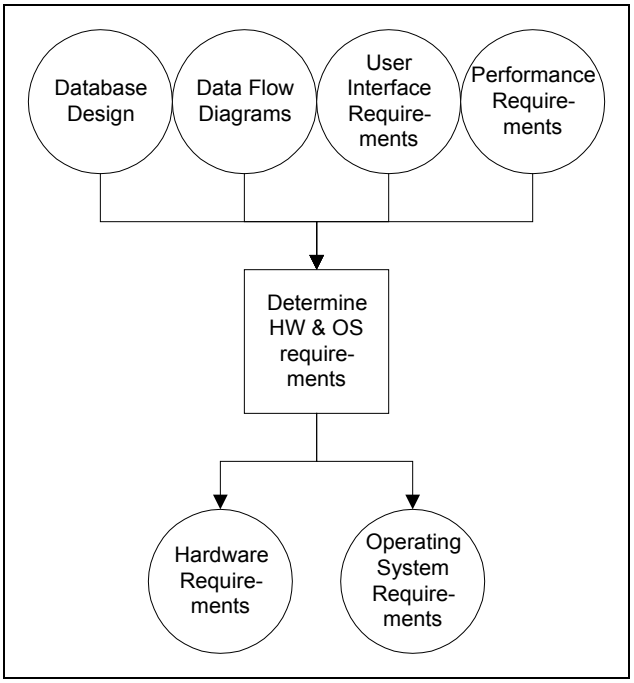


Figure 8–6. Analyze Platform and Operation System

Activity 3.1.6: Design for Functional Requirements

See Figure 8–7. Using the results of the

- Data analysis and database design
- Process analysis and design
- User interface analysis and design
- Hardware and operating system analysis

design any additional functions required to create an information management system that

- Reuses corporate databases
- Converts corporate data as required
- Provides support for automating processes appropriately
- Supports the user interfaces as designed above

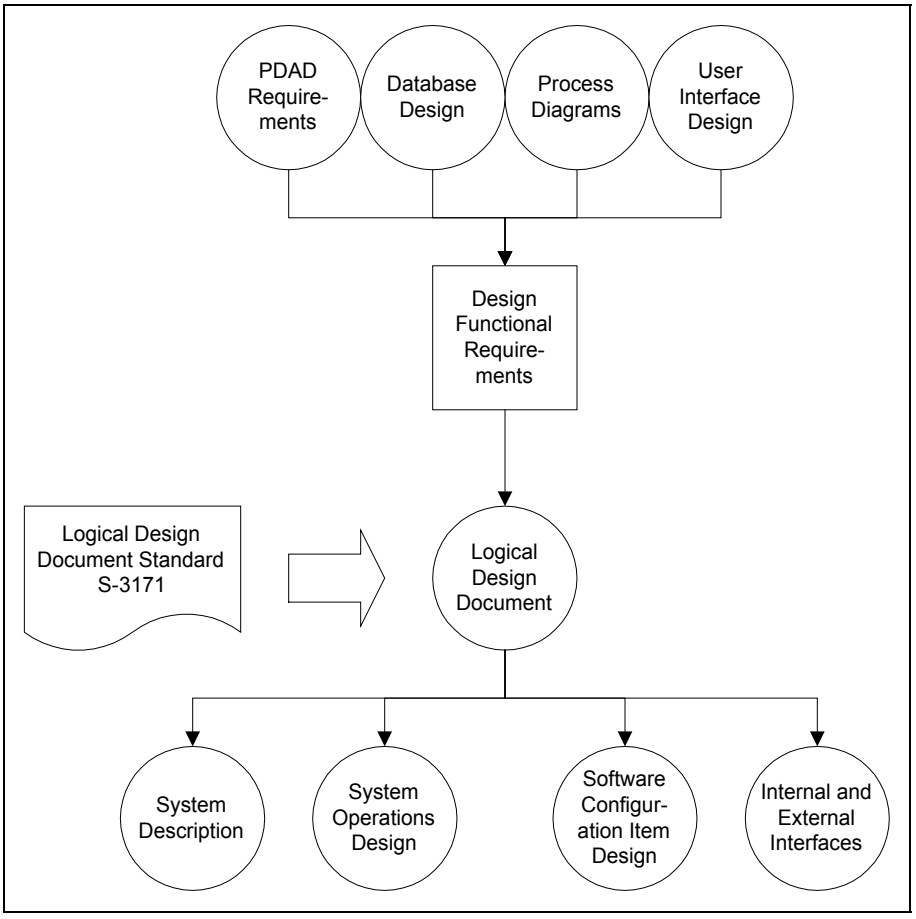


Figure 8–7. Design for Functional Requirements

Activity 3.2: Plan Solution Integration

See Figure 8–8. Using the results of Activity 3.1, develop a plan for integrating the solution.

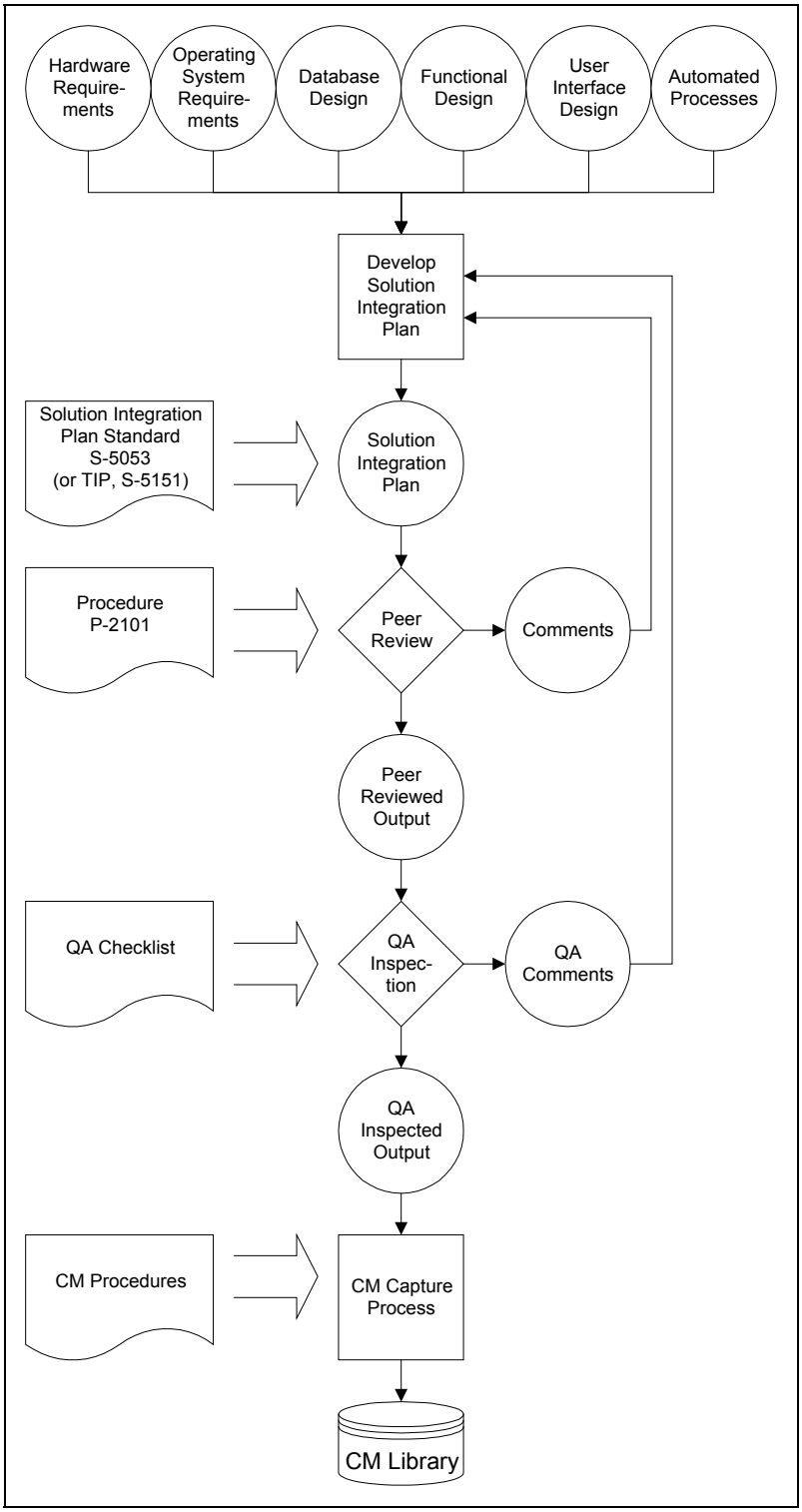


Figure 8–8. Plan Solution Integration

Activity 3.3: Design Training Materials

See Figure 8–9. Using the system requirements and the results of requirements analysis and high level design, design the training materials.

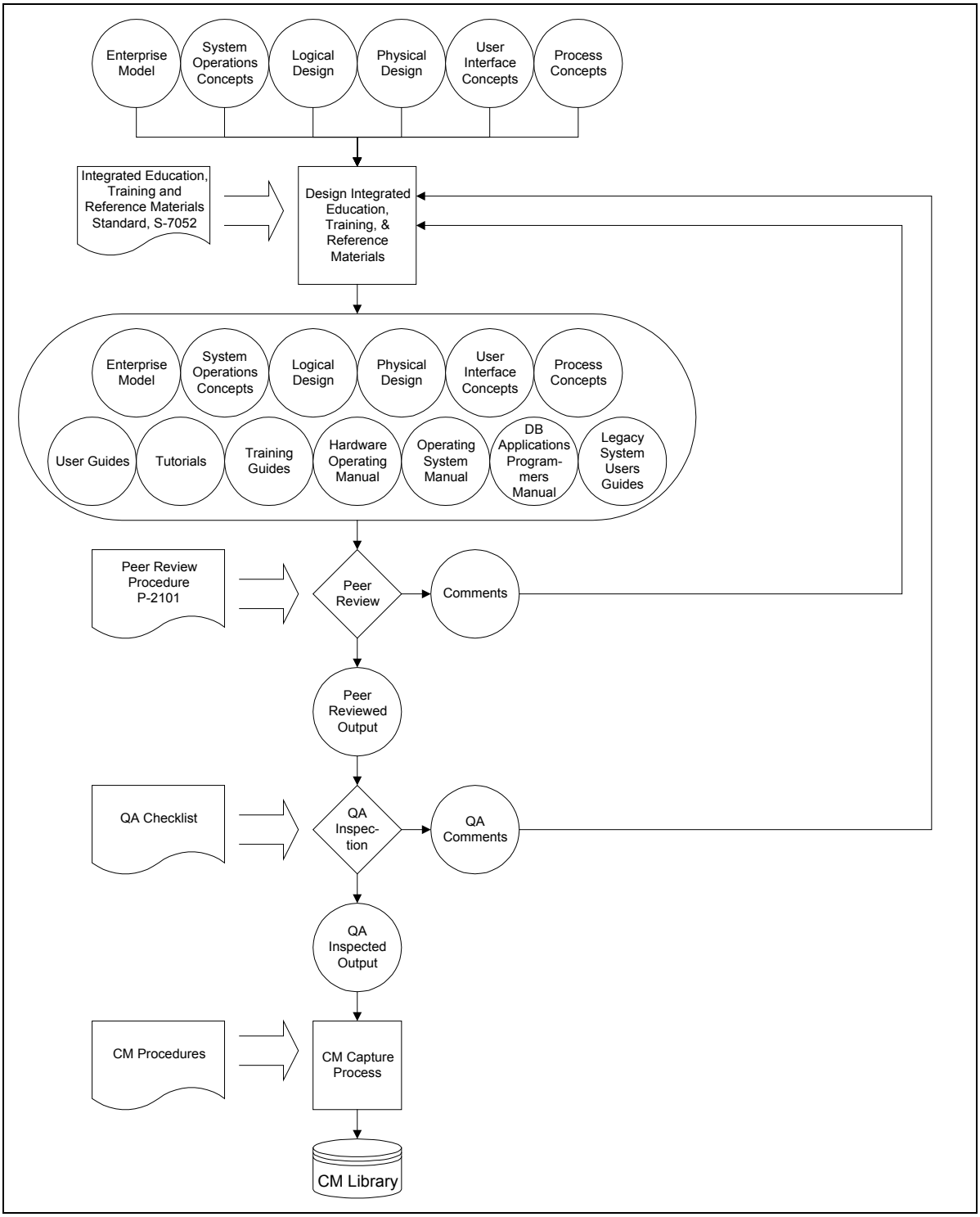


Figure 8–9. Design Training Materials

Activity 3.4: Establish Test Approach

See Figure 8–10. Using the results of the requirements analysis and high level design, including knowledge of the:

- Top-level requirements (functions)
- Systems Operations Concepts
- Systems Integration Plan
- New System Functions and Databases
- Computer Software Configuration Items (CSCIs)
- Units

specify a bottom-up test approach, which must contain:

- Unit Tests
- Modules Tests
- New Integrated Solution Test
- Systems Test
- Requirements Test

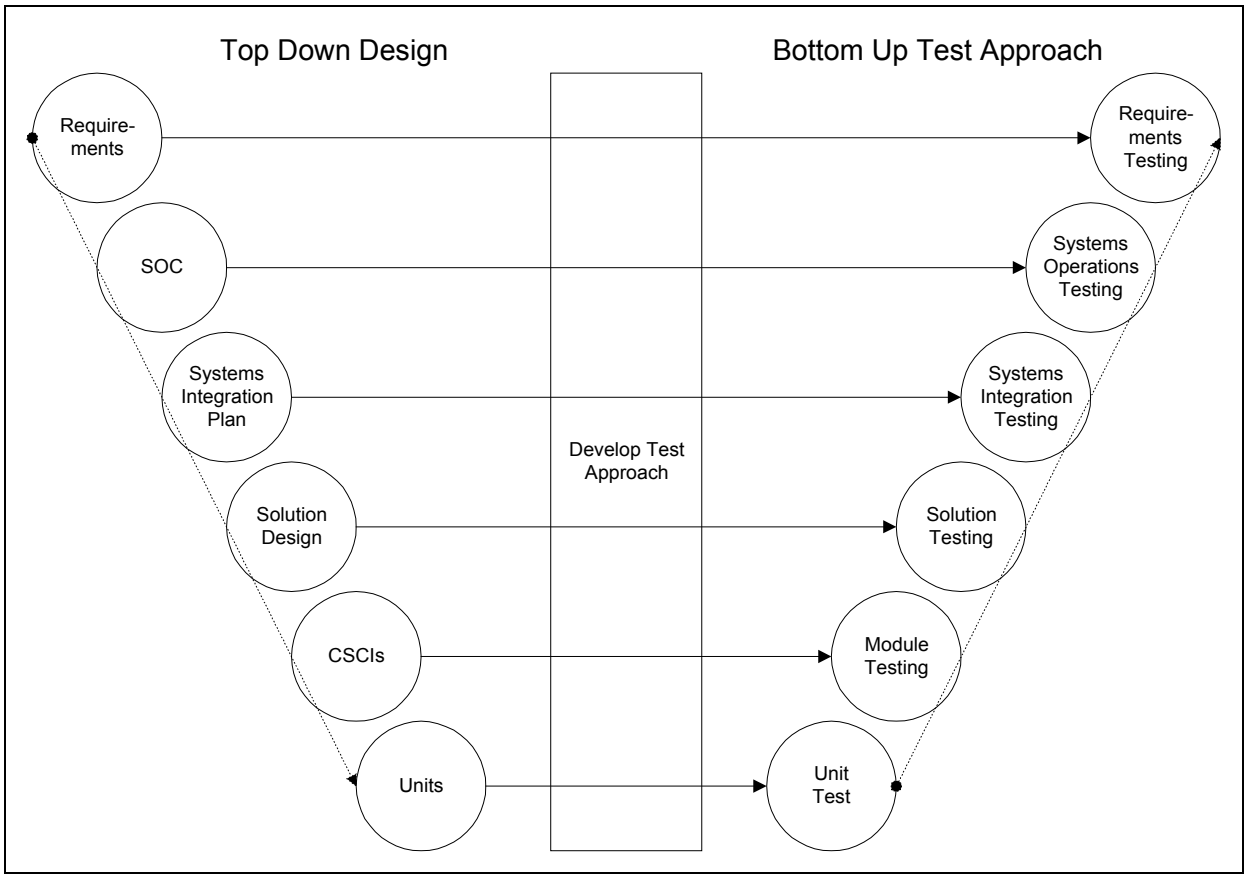


Figure 8–10. Establish Test Approach

Activity 3.5: Update Project Action Plan

See Figure 8–11. Using the work results, identify any changes required in the resources, schedules, and activities. Update the Project Action Plan.

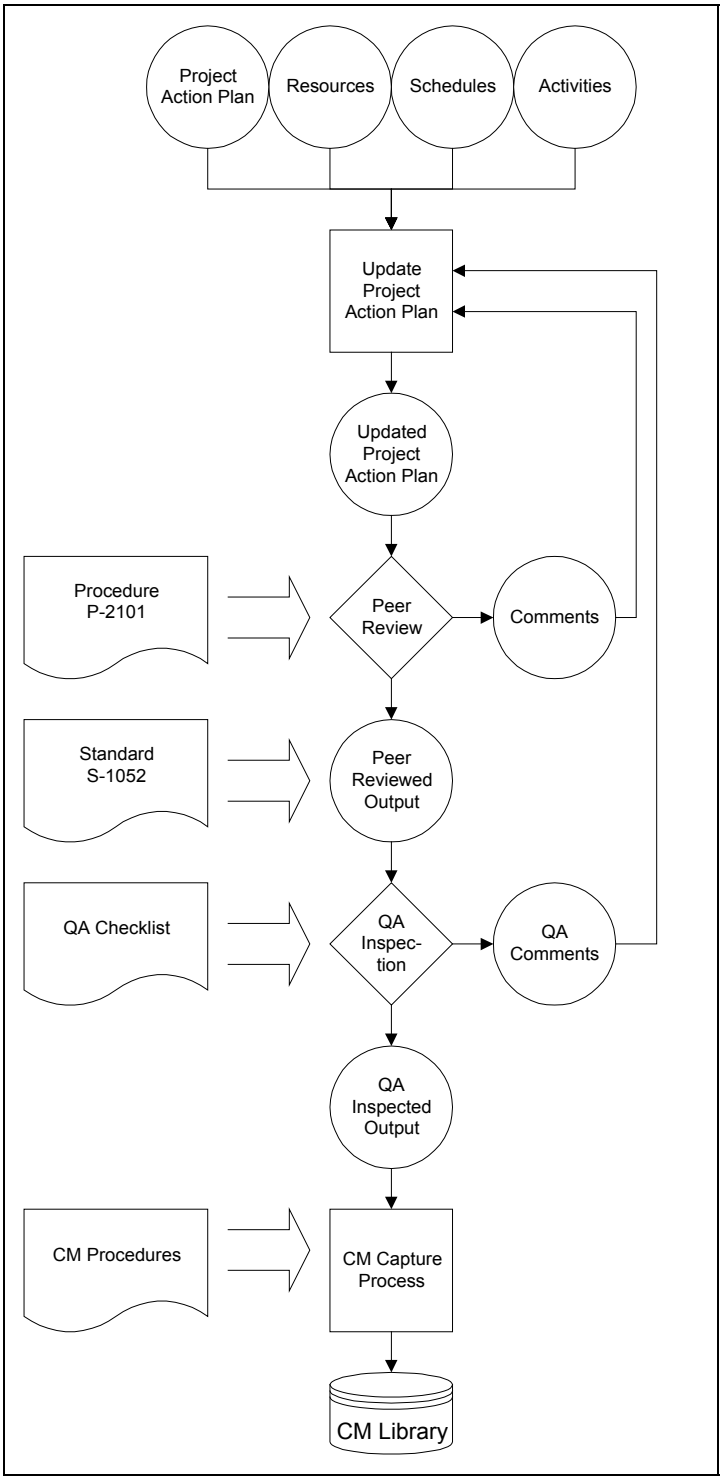


Figure 8–11. Update Project Action Plan

Activity 3.6: Continue Deployment Planning

See Figure 8–12. Using the Solution Integration Plan, and the results of the network analysis, analysis of other systems that can be reused verbatim, analysis of others systems that can be reused in part by establishing conversion requirements, establish a system integration plan.

1. Identify other systems for integration
2. Identify systems security requirements
3. Update the other portions of the TIP or development separate sub-plans
4. Update Technical Reference Model

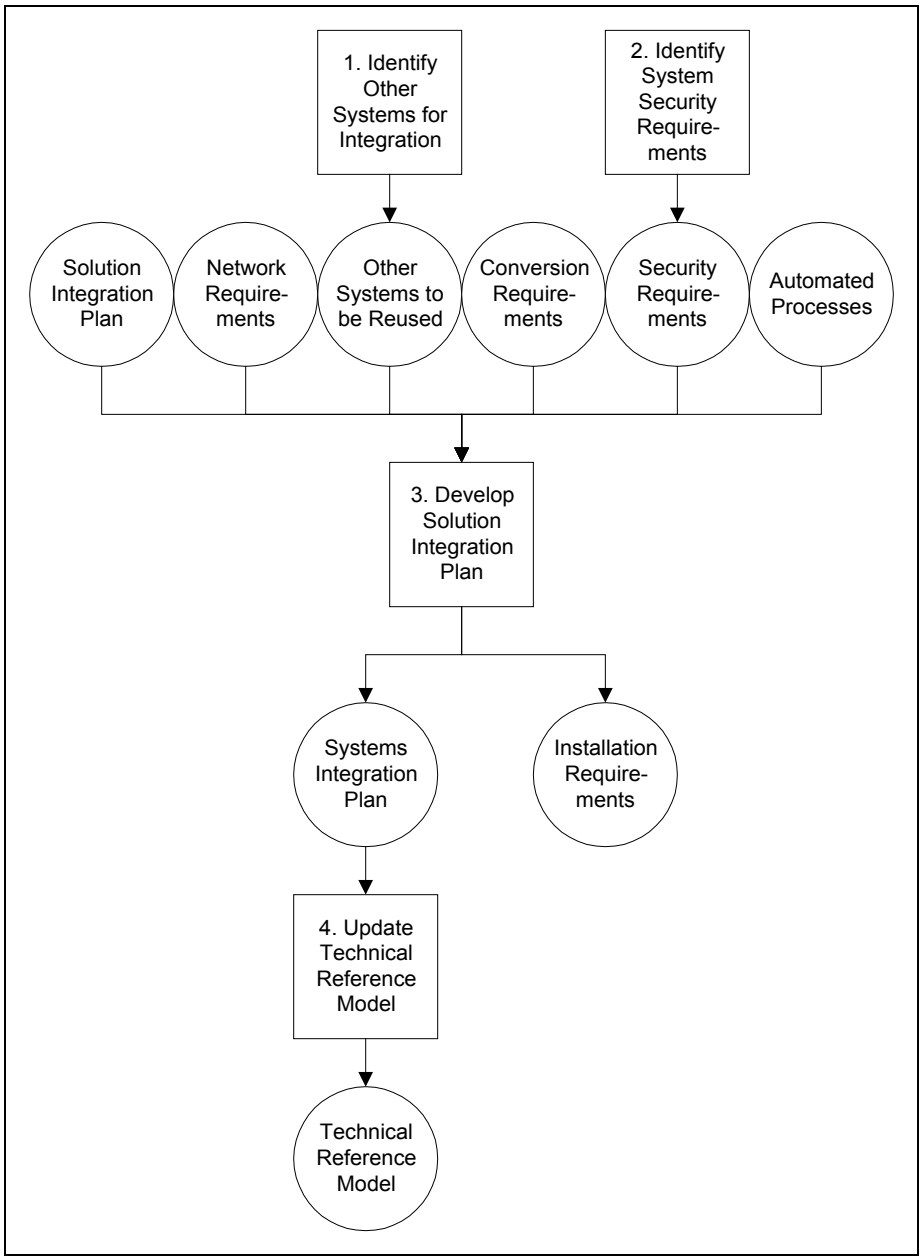


Figure 8–12. Continue Deployment Planning

Activity 3.7: Review (Component 3)

See Figure 8–13. Review all products of the project to make a GO or NO-GO decision before proceeding to engineer all or a portion of the solution. Ensure that all stakeholders, including the Records Management Branch, are included in the review activities.

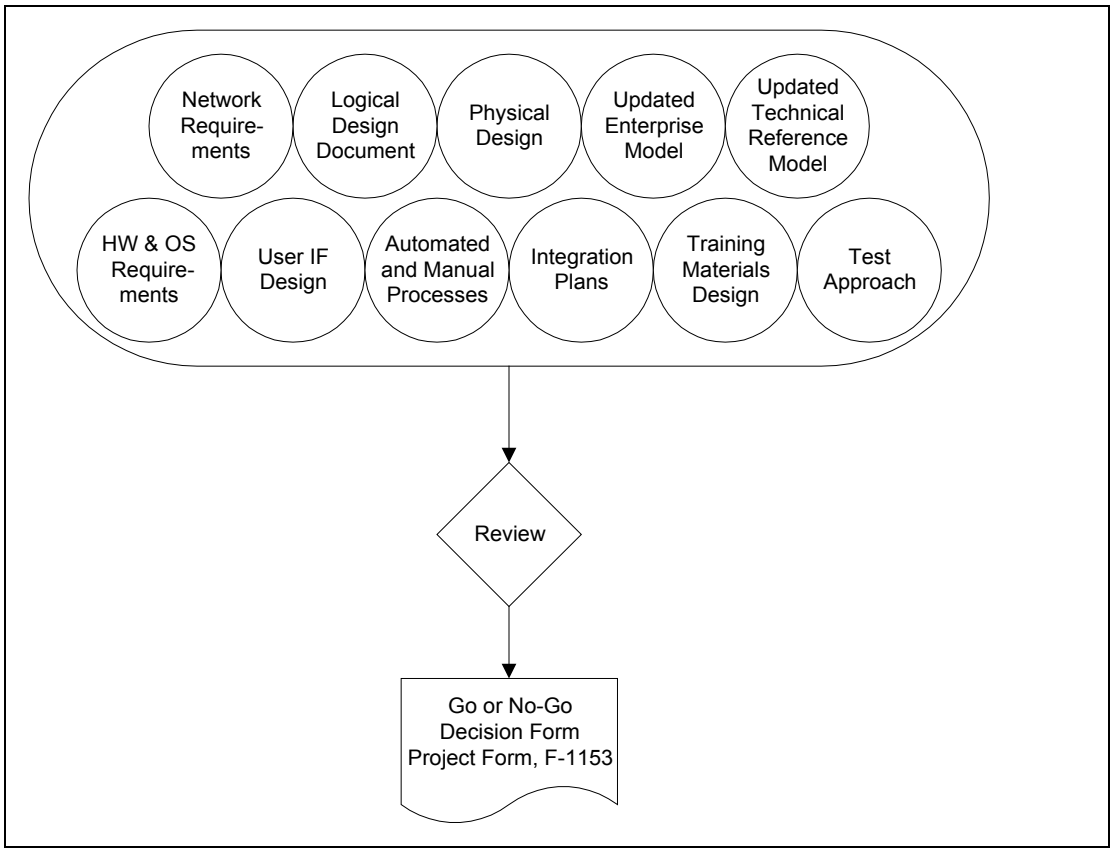


Figure 8–13. Review (Component 3)

9. Component 4. Engineer the Solution

9.1 Purpose

The purpose of this component is to build the application system, which includes the following:

- Maintain a change log
- Select an engineering approach
- Construct or acquire all modules of the system,
- Integrate the solution
- System test the solution
- Create the rollout strategy and continue planning for deployment
- Construct the training and education materials
- Update the Project Action Plan
- Secure a Go decision to deploy the project

9.2 Roles and Responsibilities

The roles required to perform the activities in the Engineer the Solution component are shown in Table 9–1.

Table 9–1. Component 4 Roles and Responsibilities

Roles	Responsibilities
Overall Project Manager Technical Project Manager Development Team	Accountable for Execution
Business Advocate	Approval to Proceed
Executive Sponsor Business Project Manager Business SME Technical SME Other Representatives or Stakeholders • Quality Assurance Manager • Configuration Management Manager	Provides Input

9.3 Entry Criteria

The following event may trigger the initiation of Component 4:

- Some portion of the solution design approved for engineering

9.4 Input, Activities, and Outputs

The inputs to this component are as follows:

- Updated Project Action Plan
- Updated Project Definition and Analysis Document
- Logical Design Document
- Physical Design Document
- Updated Tactical Integration Plan, including
  - Products Installation and Integration Plan
  - Other Integration Issues Plan
  - Solution Integration Plan
  - Other Systems Integration Plan
  - Security Plan
  - Conversion Plan
- Test Plan
- Integrated Education, Training, and Reference Materials
- Software Engineering Notebook
- Change Requests, if any

Figure 9–1 illustrates the flow of the following activities:

1. Maintain the Change Log
2. Engineer the Detailed Design
3. Build the Solution
4. Integrate the Solution
5. Test the Solution
6. Create the Rollout Strategy and Continue Deployment Planning
7. Build the User Material
8. Update the Project Action Plan
9. Review (Component 4)

The inputs to this component were produced as products of earlier components.

As the figure suggests, many of the activities can be performed in parallel. The activities are described in more detail in Section 9.7.

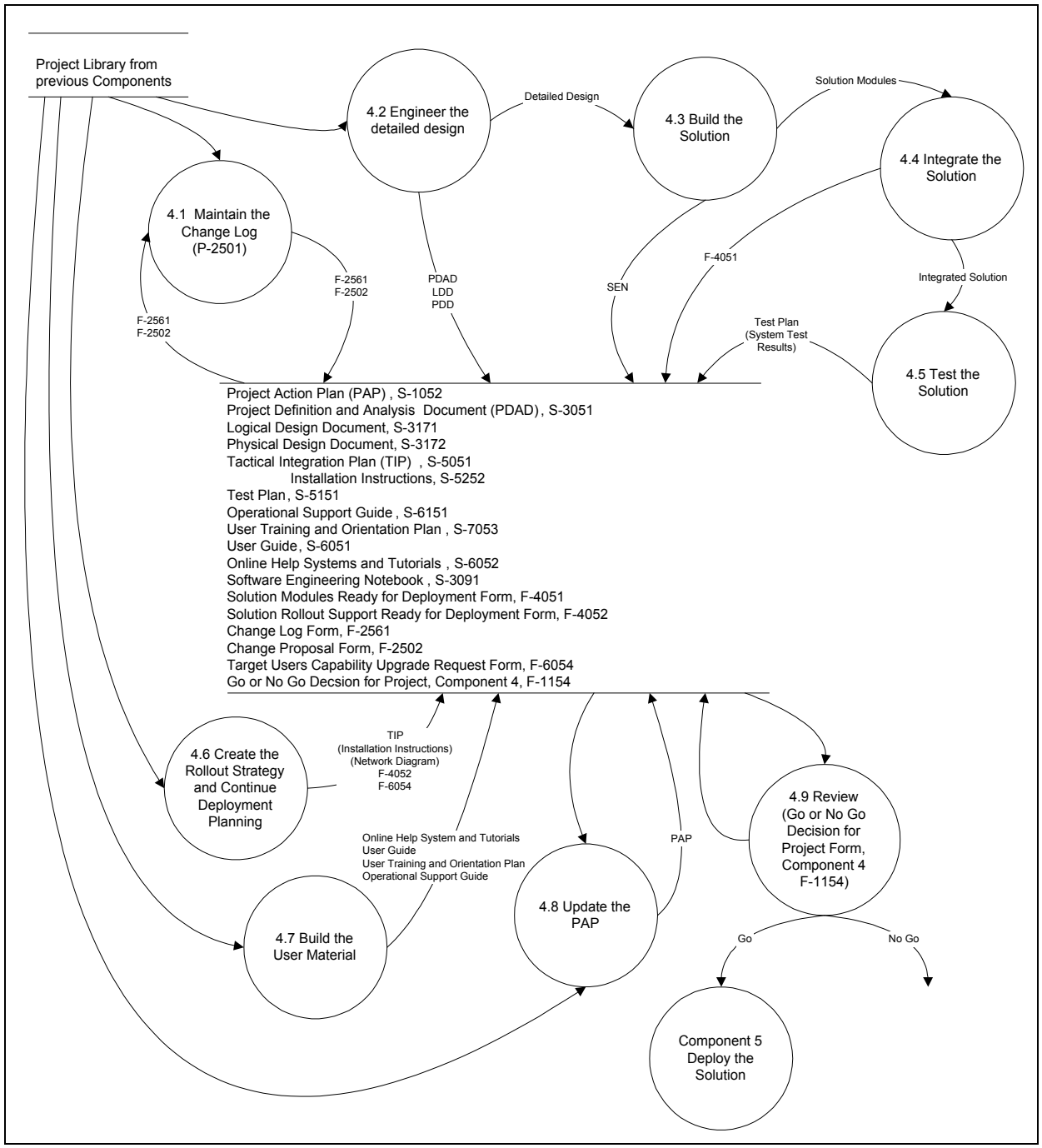


Figure 9–1. Component 4

Figure 9–1 also summarizes the outputs of all Component 4 activities in the central data store.

Appendix C includes a summary description of all products of NRC projects, indicates within which components each product is created and updated, and specifies which products are required.

The products, whose standard and form numbers are shown in the figure, are described in more detail in the companion volume *SDLCM Methodology Procedures, Standards, and Forms*.

9.5 Techniques and Tools

The activities of Component 4 are supported by the following techniques.

- Joint Application Design and Development
- Rapid Application Design
- Structured Facilitation
- Structured Walkthrough
- Peer Review

Refer to the current *SDLCM Methodology Tool Inventory* for the recommended set of tools.

9.6 Exit Criteria

Component 4 is complete when:

- All critical products have been reviewed (structured walkthrough or peer review)
- Key products have been inspected by QA to conform to project standards
- Products have been placed under CM control
- Information has been shared
- Issues have been resolved
- There is management buy-in to move forward
- There are no reasons to stop this project now

9.7 Component 4 Activity Details

The following pages provide detailed activities of Component 4, Engineer the Solution.

Activity 4.1: Maintain the Change Log

See Figure 9–2. When any portion of the design is approved for engineering, it is important to maintain a record of any proposed changed (whether required or requested).

1. Initiate the change log. Use SDLCM Methodology Form F–2561 or an automated software tool that provides at least the same information.
2. Process change proposals and maintain the change log following the Change Proposal procedure (P–2502) and using the Change Proposal Form (F–2502).

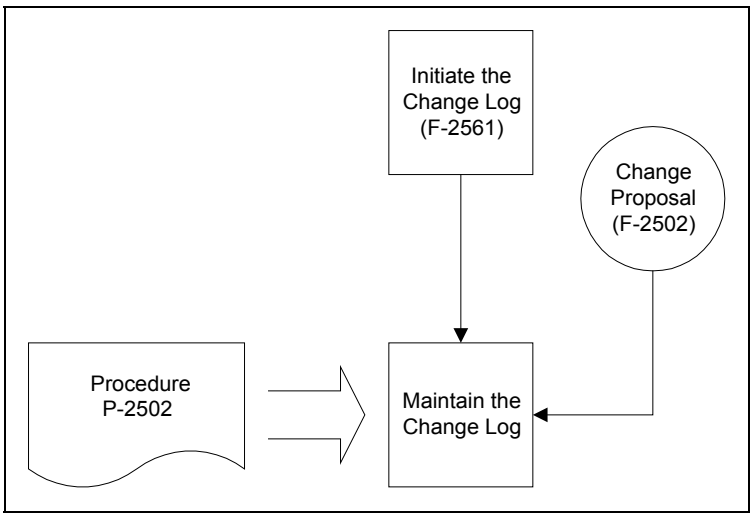


Figure 9–2. Maintain the Change Log

Activity 4.2: Engineer the Detailed Design

See Figure 9–3. Given the scope of the work to be done as documented in the PDAD, Design documents, and Project Charter, and considering the previous analysis of alternatives and final System Operations Concept that is part of the PDAD, select an engineering approach that both maximizes reuse and the probability of success.

Perform the following sub-activities:

1. Determine if there is an off-the-shelf solution that can be directly installed
2. If not, determine if there is an off-the-shelf solution that can be modified and installed.
3. If not, plan to develop the system from scratch.
4. Using a top-down approach, partition the system into modules, specifying which parts can be reused or modified or built from scratch.
5. Review the life-cycle approach selected and schedule the development and integration of the modules and document this in the Project Action Plan.

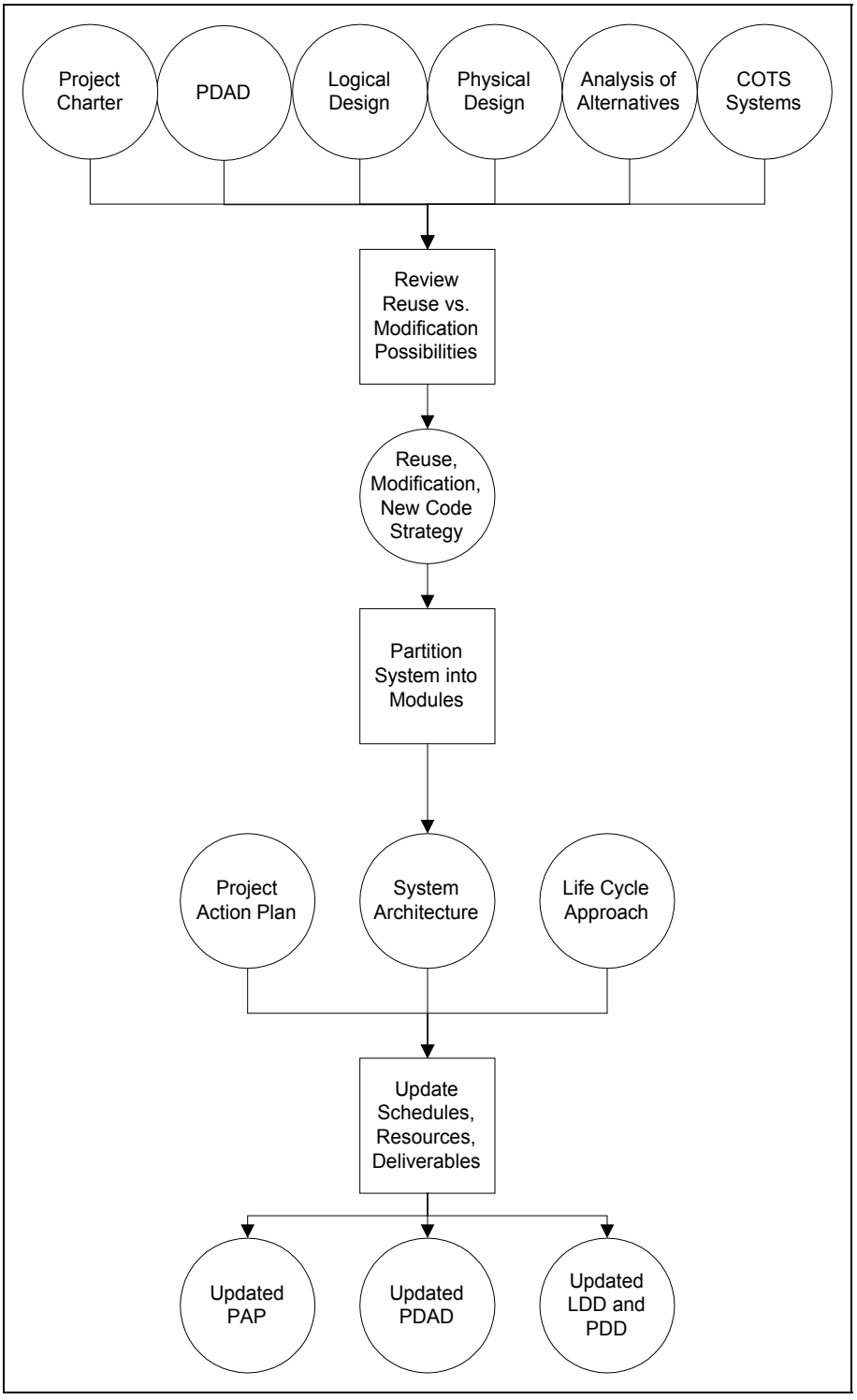


Figure 9–3. Engineer the Detailed Design

Activity 4.3: Build the Solution

See Figure 9–4, Using the design materials produced as a part of Component 3 and earlier activities within Component 4,

1. Build database structure and populate the database
2. Develop program code
3. Develop external systems interfaces

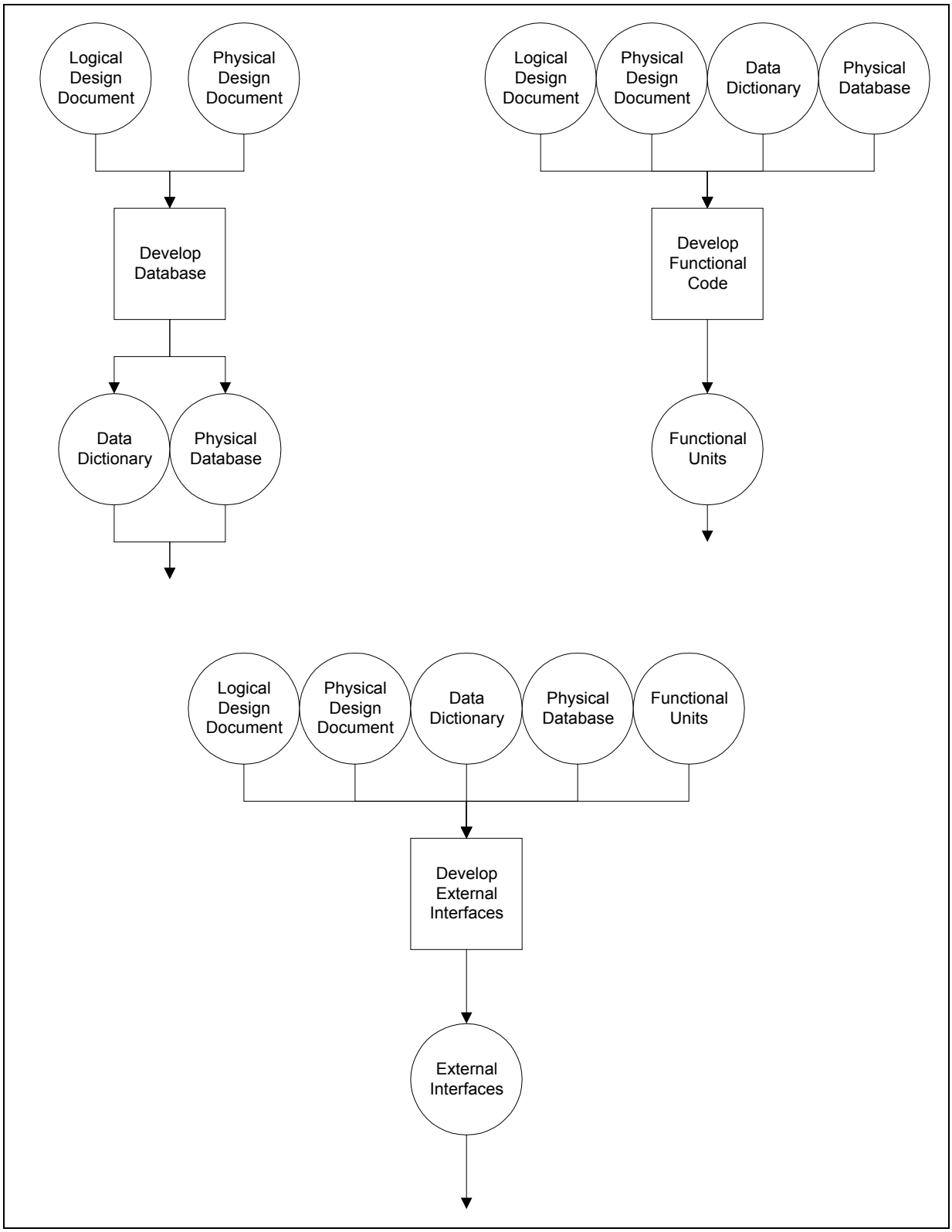


Figure 9–4. Build the Solution

Activity 4.4: Integrate the Solution

See Figure 9–5. As parts of a solution are built, they must be integrated into larger and larger parts until the system is built and integrated into the overall enterprise. Integrate the system in a step-wise fashion as follows:

1. Integrate units into modules.
2. Integrate modules into systems.
3. Integrate the systems into the network of systems, that is, the enterprise.

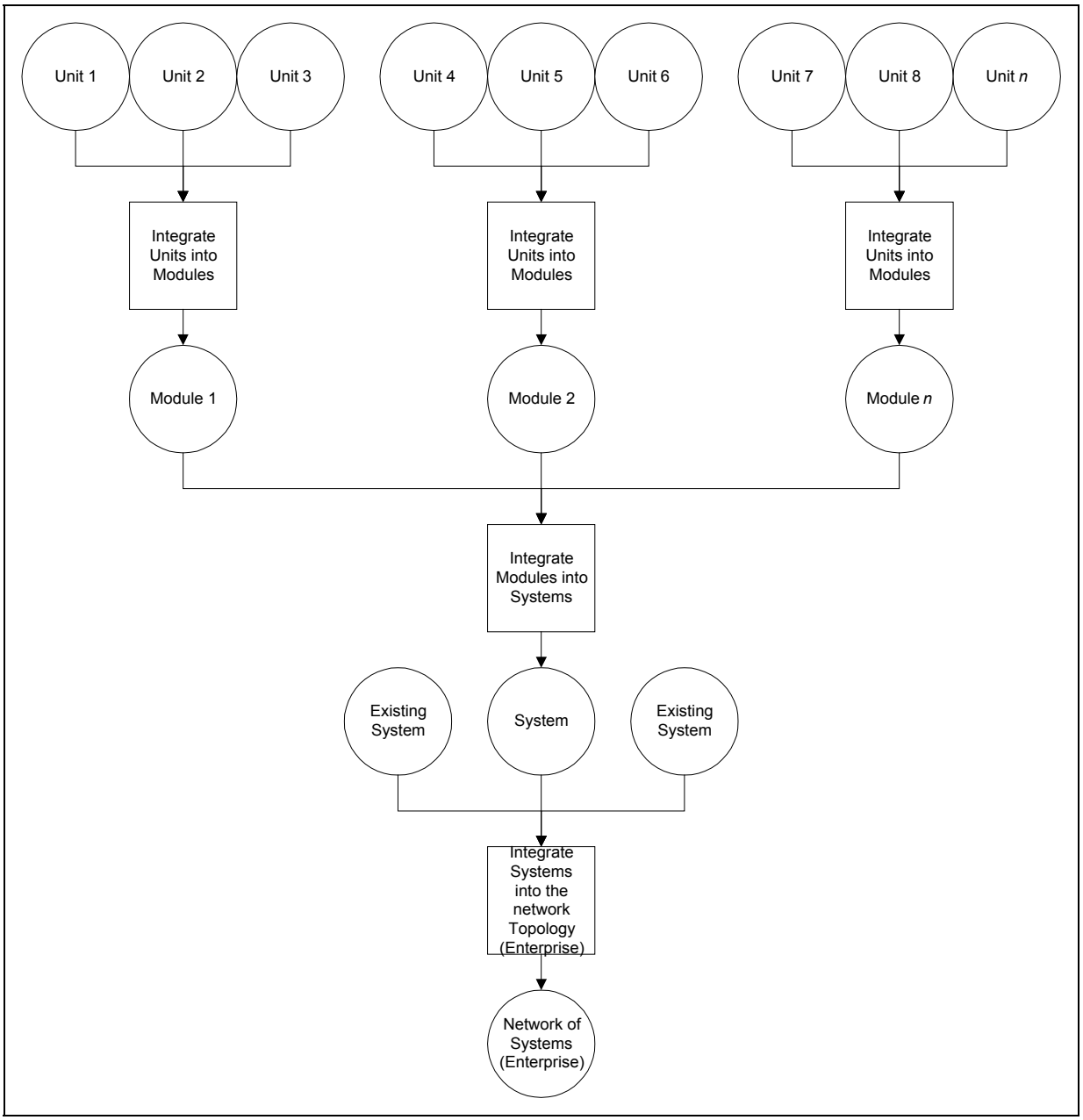


Figure 9–5. Integrate the Solution

Activity 4.5: Test the Solution

See Figure 9–6. As each part of the solution is built, it should be tested before being integrated into a larger part that is also tested. Just as the Build the Solution activity of the Engineer the Component is bottom up, so is the testing activity.

For each kind of test:

- Unit
- Module
- System
- Acceptance

Perform each of the following kinds of sub-activities:

- Develop test data.
- Develop a test plan (that is, a test scenario or execution plan that specifies the inputs and activities required for performing the test).
- Document the expected results.
- Execute the test and record results.
- Analyze the test results. Compare the actual results with the expected results.
- If the test results agree with the anticipated results, then enter the component into the configured library.
- If the test results do not agree with the anticipated results, generate a problem report if applicable, investigate the cause, and generate appropriate change requests to fix the source of the problem.

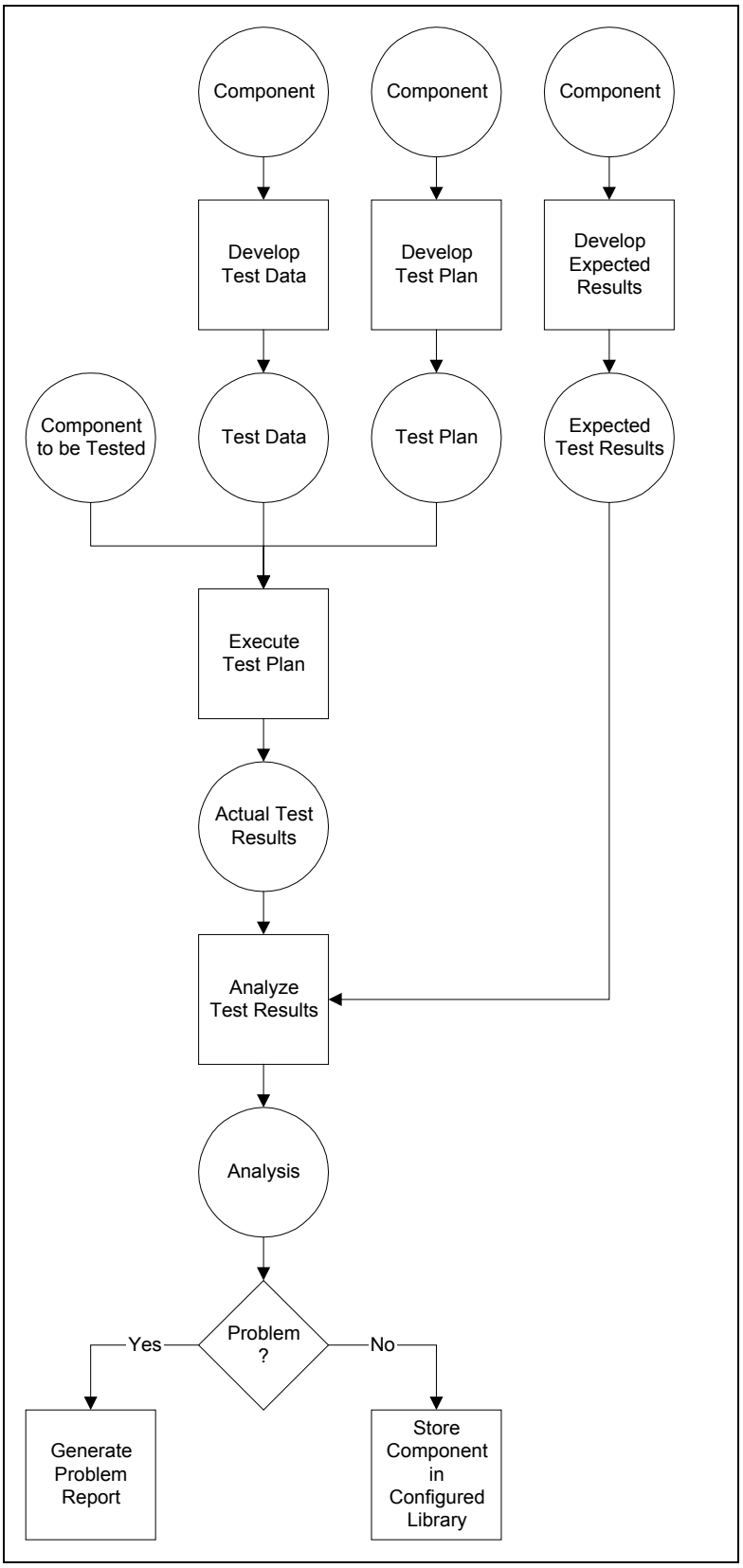


Figure 9–6. Test the Solution

Activity 4.6: Create the Rollout Strategy and Continue Deployment Planning

See Figure 9–7. Given the locations and organizations that will receive the new information management system, develop a rollout strategy and document this in the Tactical Integration Plan.

1. Define the target users
2. Survey the target users desktops to determine their capabilities and needs for upgrades
3. Create a schedule for the upgrades
4. Create a schedule for the installation of the new system
5. Package the Rollout Strategy as part of the Tactical Integration Plan.

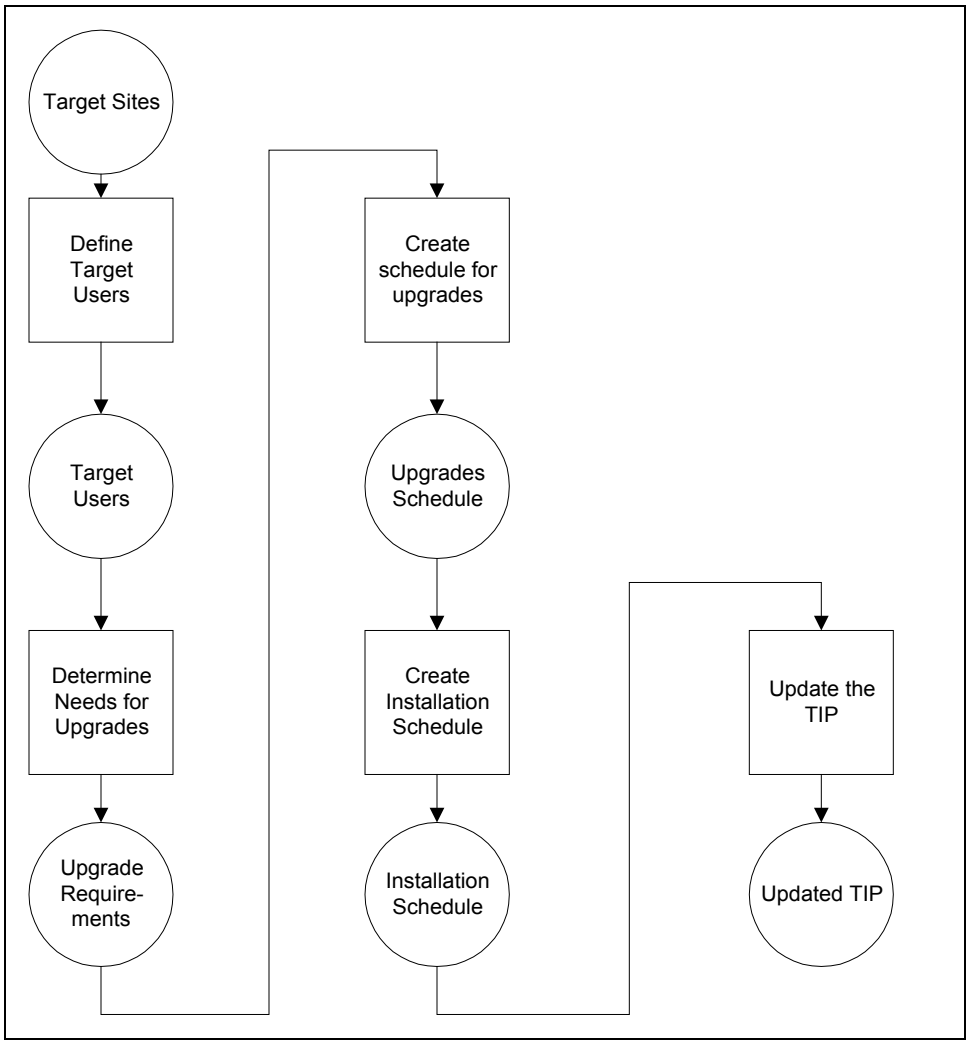


Figure 9–7. Create the Rollout Strategy and Continue Deployment Planning

Activity 4.7: Build the User Material

See Figure 9–8. Using (a) the plan for building an integrated set of education, training, and support materials from the Design the Solution component, (b) the design, and (c) the actual system specifications, build the materials to support the users.

- Produce user guides
- Produce a help system
- Produce operational guides
- Produce tutorials
- Create and maintain a customer support center
- Develop a user training plan

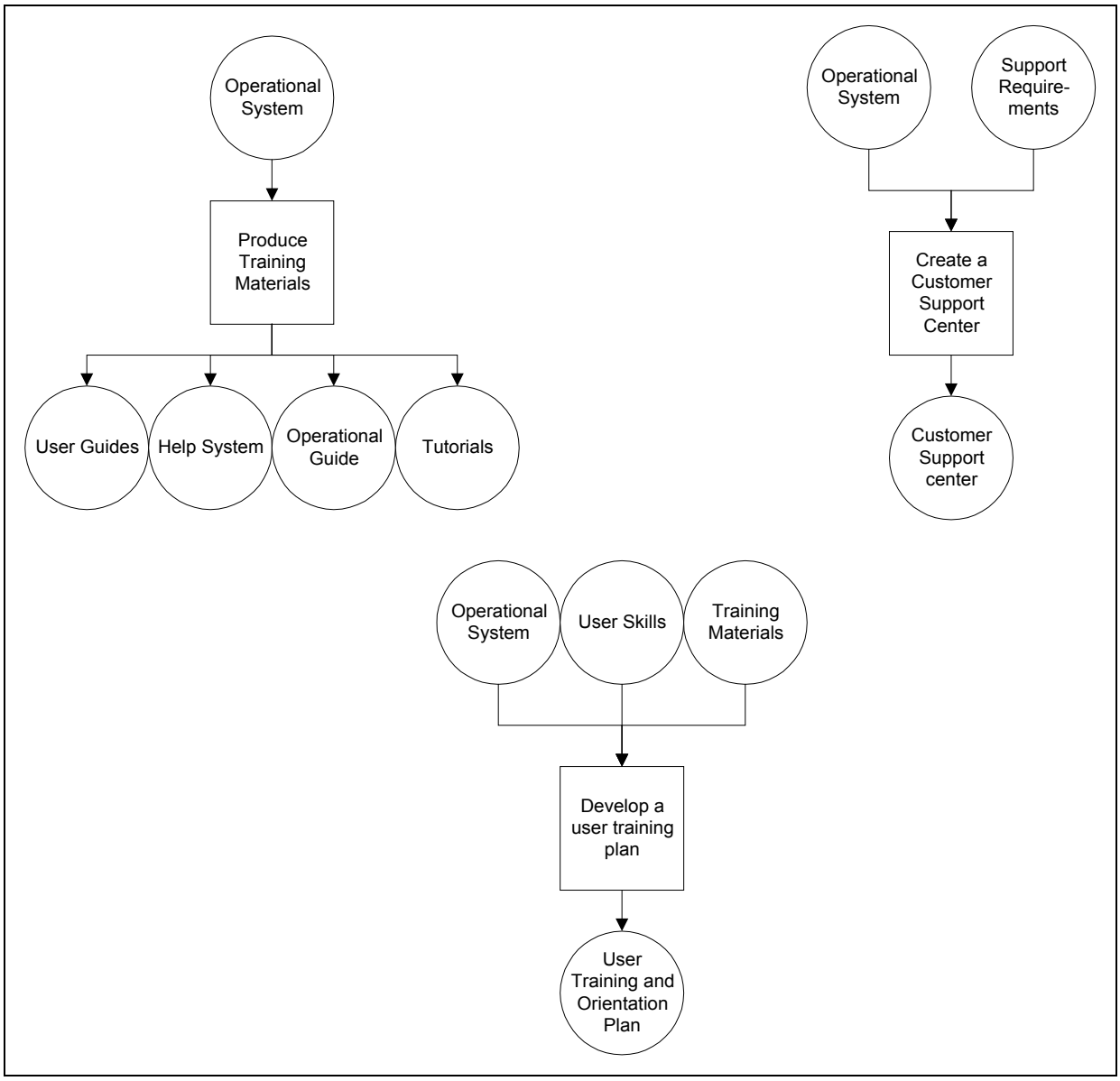


Figure 9–8. Build the User Material

Activity 4.8: Update the Project Action Plan

See Figure 9–9. Prepare for the next component by updating the Project Action Plan with information produced during this component.

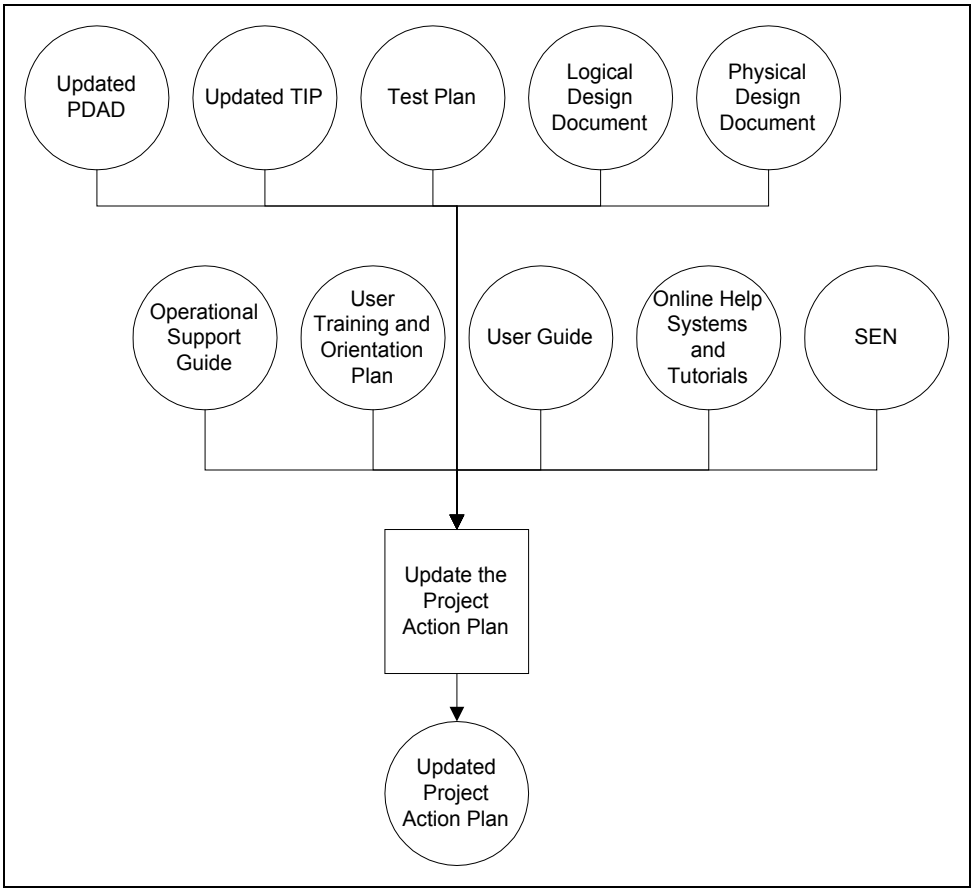


Figure 9–9. Update the Project Action Plan

Activity 4.9: Review (Component 4)

See Figure 9–10. Review the results of the activities that were performed in this component to determine if the system should be deployed. Include the Records Management organization in this review activity.

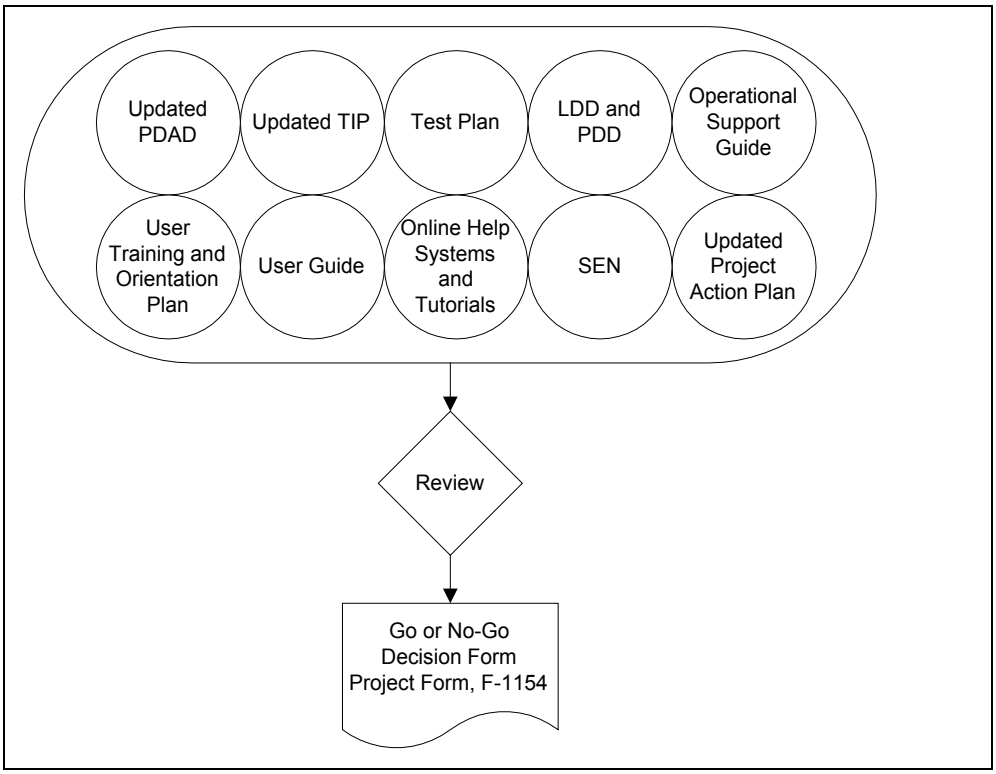


Figure 9–10. Review (Component 4)

10. Component 5. Deploy the Solution

10.1 Purpose

The purpose of this component is to roll out the integrated solution, which includes:

- Validate the environment and update it if required
- Install the solution
- Test the installed solution
- Training the users
- Acceptance test the system
- Review the results of the deployment and test
- Prepare feedback reports
- Obtain user approval
- Cut over to a production environment
- Use the system
- Update the technical inventory

10.2 Roles and Responsibilities

The roles required to perform the activities in the Deploy the Solution component are shown in Table 10–1.

Table 10–1. Component 5 Roles and Responsibilities

Roles	Responsibilities
Overall Project Manager Technical Project Manager Development Team	Accountable for Execution
Business Advocate	Approval to Proceed
Executive Sponsor Business Project Manager Business SME Technical SME Other Representatives or Stakeholders • Quality Assurance Manager • Configuration Management Manager	Provides Input

10.3 Entry Criteria

The following set of events trigger the initiation of Component 5:

- System test completed
- Rollout plan completed
- Training plan and materials completed
- Installation instructions completed
- Regression test completed
- Customer acceptance

10.4 Input, Activities, and Outputs

The inputs to this component are as follows:

- Project Action plan
- Tactical Integration Plan, including
  - Conversion plan
  - Rollout plan
  - Training plan and materials
  - Installation instructions
  - Performance criteria measures

Figure 10–1 illustrates the flow of the following activities:

1. Review and Update Plans and Announce Deployment
2. Validate and Upgrade the Environment
3. Install the Solution
4. Test the Installed Solution
5. Analyze the Test Results
6. Train the Users
7. Acceptance Test the Solution
8. Cut over to Production Environment
9. Update Policies and Procedures
10. Review (Component 5)

The inputs to this component were produced as products of Component 4.

As the figure suggests, many of the activities can be performed in parallel. The activities are described in more detail in Section 10.7.

Figure 10–1 also summarizes the outputs of all Component 5 activities in the central data store.

Appendix C includes a summary description of all products of NRC projects, indicates within which components each product is created and updated, and specifies which products are required.

The products, whose standard and form numbers are shown in the figure, are described in more detail in the companion volume *SDLCM Methodology Procedures, Standards, and Forms*.

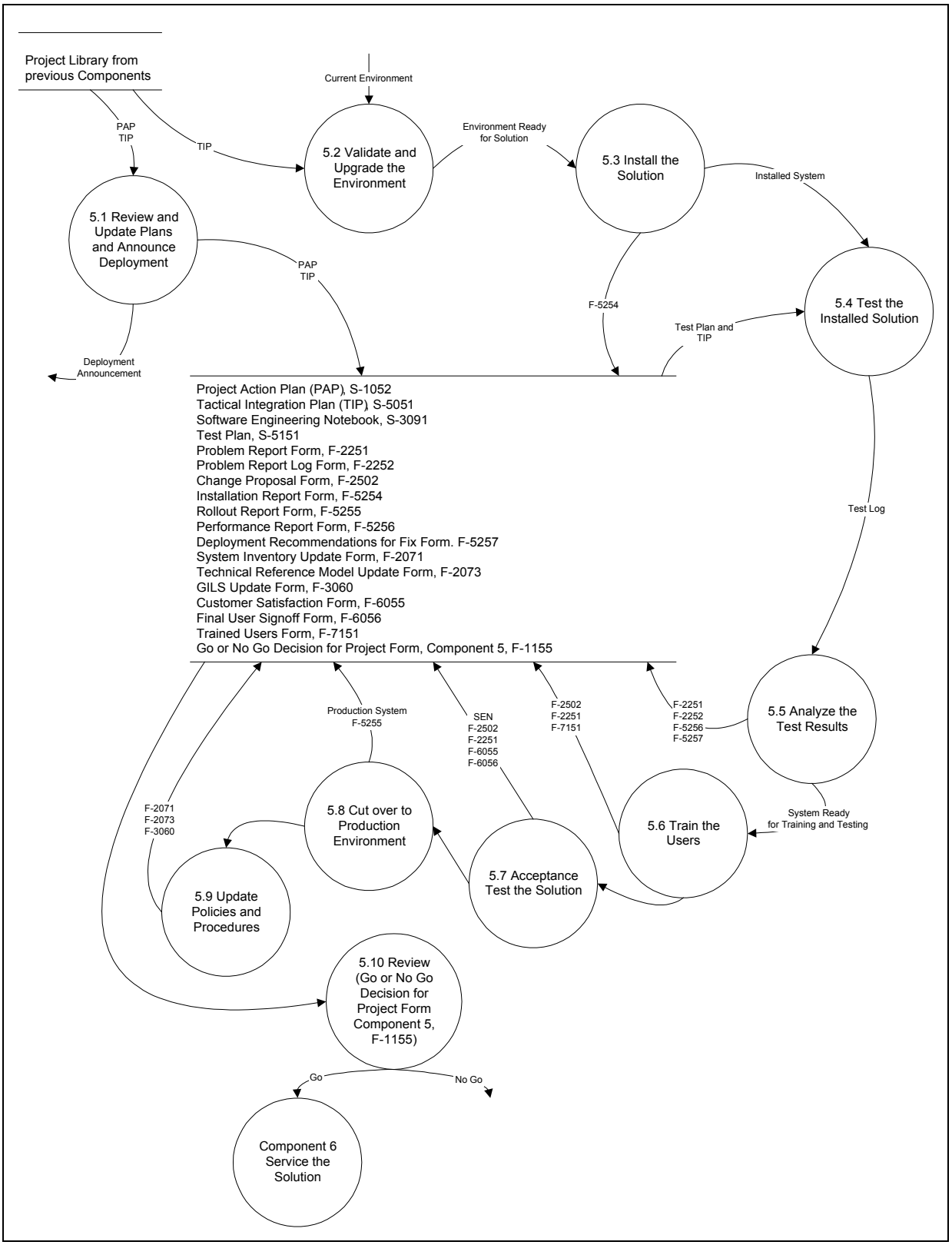


Figure 10–1. Component 5

10.5 Techniques and Tools

The activities of Component 5 are supported by the following techniques.

- Structured Facilitation
- Structured Walkthrough
- Peer Review

Refer to the current *SDLCM Methodology Tool Inventory* for the recommended set of tools.

10.6 Exit Criteria

Component 5 is complete when:

- All critical products have been reviewed (structured walkthrough or peer review)
- Key products have been inspected by QA to conform to project standards
- Products have been placed under CM control
- Information has been shared
- Issues have been resolved
- There is management buy-in to move forward
- There are no reasons to stop this project now

10.7 Component 5 Activity Details

The following pages provide detailed activities of Component 5, Deploy the Solution.

Activity 5.1: Review and Update Plans and Announce Deployment

See Figure 10–2. Review the Project Action Plan and TIP, and announce the deployment. It is important to prepare people to expect deployment activities

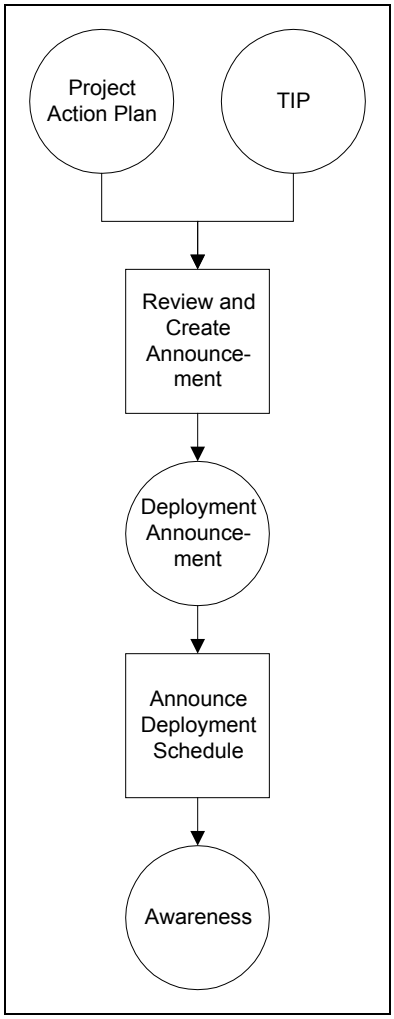


Figure 10–2. Review and Update Plans and Announce Deployment

Activity 5.2: Validate and Upgrade the Environment

See Figure 10–3. Using the TIP,

1. Validate the environment (workstation and host or client and server) to determine what the system needs to run and whether the system will run in each individual user environment.
2. If needed, procure and install any additional hardware and software.
3. Upgrade the environment.

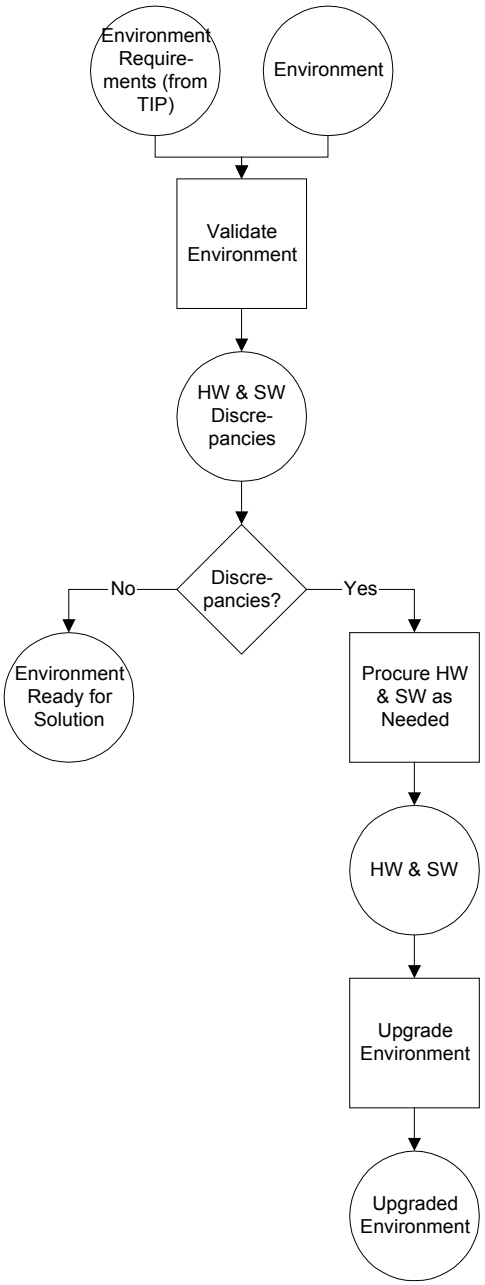


Figure 10–3. Validate and Upgrade the Environment

Activity 5.3: Install the Solution

See Figure 10–4. After the environments are validated,

- Install the solution, including the solution modules and database, using the installation instructions.
- Capture the results of the installation in an installation report (Form F–5254).

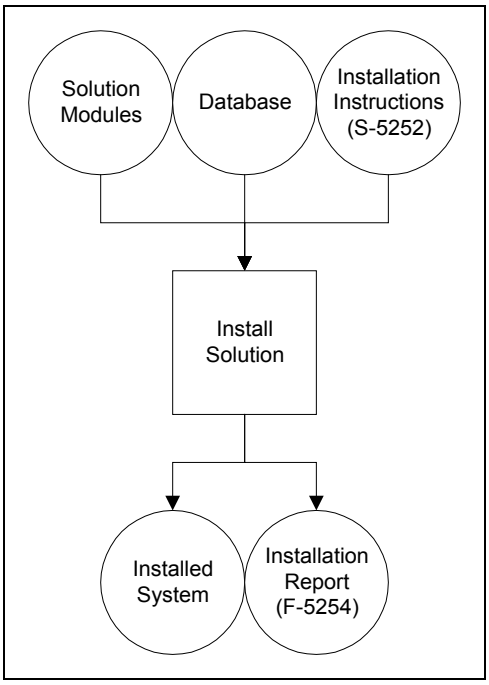


Figure 10–4. Install the Solution

Activity 5.4: Test the Installed Solution

See Figure 10–5. After the solution is installed,

1. Test it against performance specifications and functional requirements.
2. If there are any discrepancies, log them in a problem report log, and resolve the issues in accordance with project procedures.

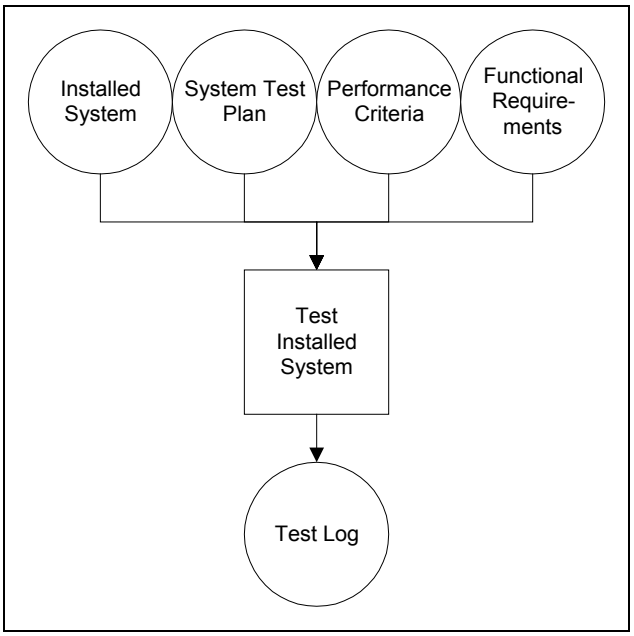


Figure 10–5. Test the Installed Solution

Activity 5.5: Analyze the Test Results

See Figure 10–6. Using the results of the system test,

1. Review the test results. Log your analysis in a feedback report.
2. If there are discrepancies, log them as problem reports (Forms F–2251 and F–2252) to be resolved in accordance with proper procedures.
3. If there are no discrepancies, then the system is ready to be used to train the users and to be acceptance tested by the customer.

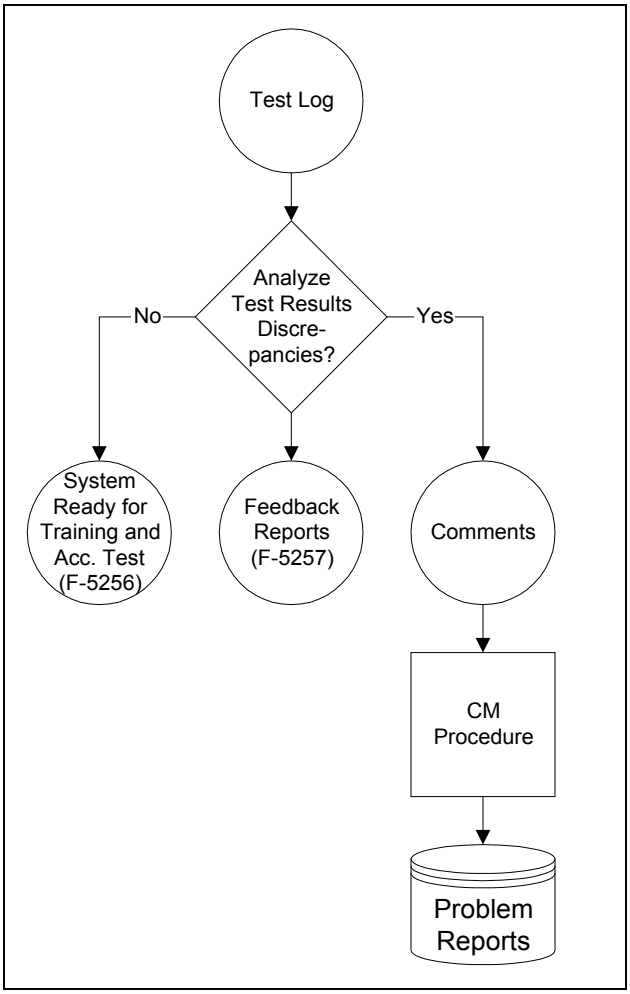


Figure 10–6. Analyze the Test Results

Activity 5.6: Train the Users

See Figure 10–7. Using the training plan and training materials prepared during the Engineer the Solution component,

1. Train the users (F–7151).
2. Generate any needed Problem Reports (F–2251) or Change Proposals (F2502).

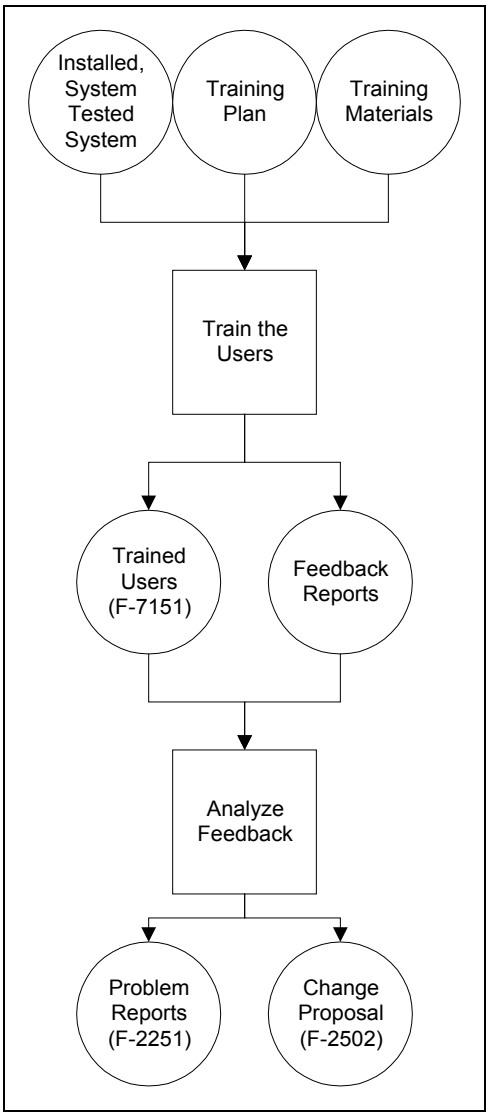


Figure 10–7. Train the Users

Activity 5.7: Acceptance Test the Solution

See Figure 10–8. After the system has been installed, any bugs have been eliminated, system testing has been completed, and a cadre of users has been trained, then the system is ready for customer acceptance testing.

1. Obtain user approval for technical environment, performance against criteria, cohabitation (test system performance).
2. Collect customer satisfaction measures (F–6055).
3. Conduct review and process any new discrepancies (F–2251 or F2502).
4. Obtain final user sign-off (F–6056).

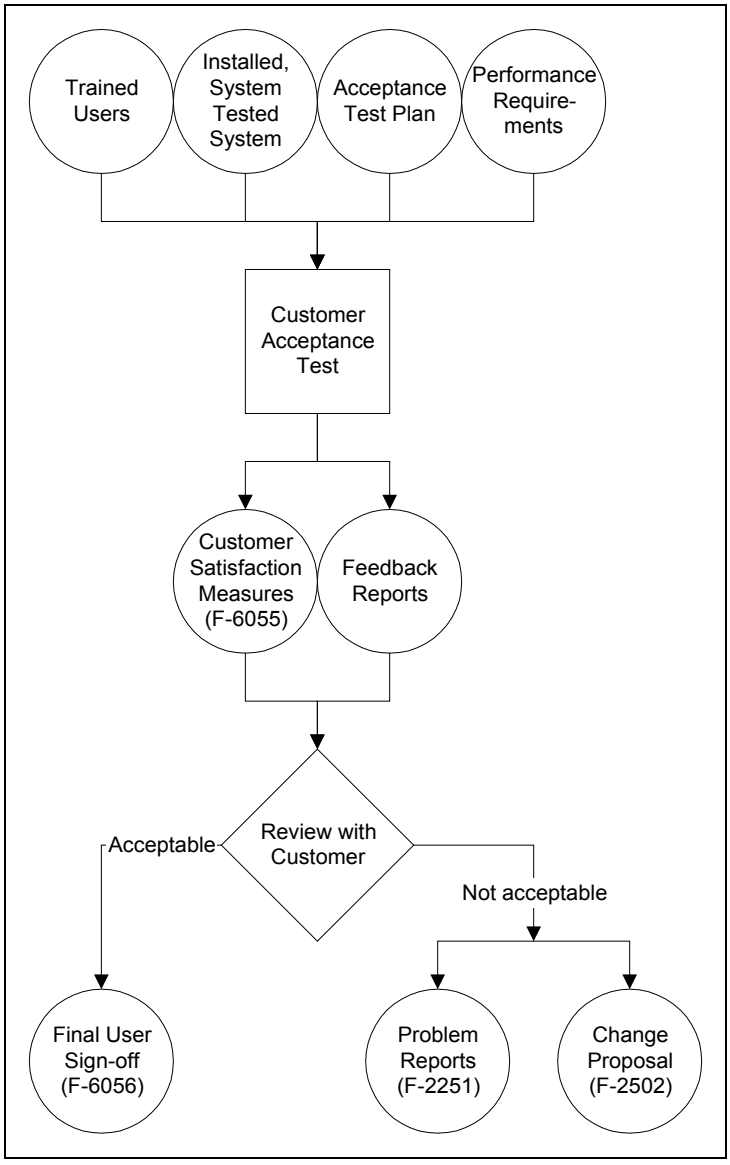


Figure 10–8. Acceptance Test the Solution

Activity 5.8: Cut over the Production Environment

See Figure 10–9. After the system has been acceptance tested and approved,

1. Provide it for operational use in the production environment.
2. Prepare a Rollout Report (F–5255).

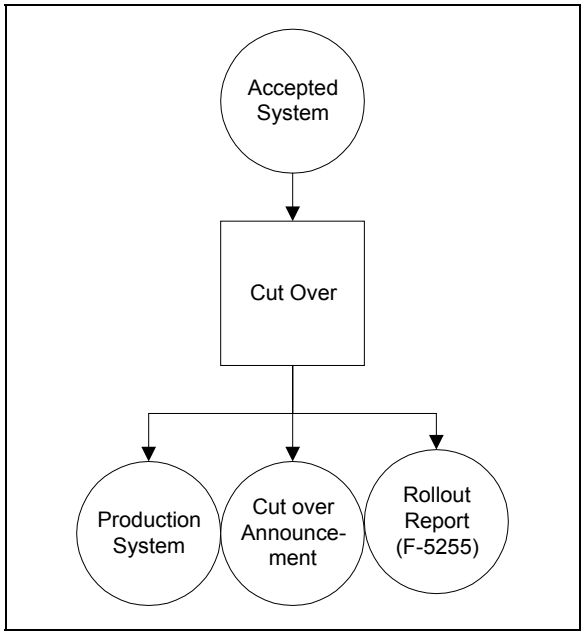


Figure 10–9. Cut over the Production Environment

Activity 5.9: Update Policies and Procedures

See Figure 10–10. After the system has been installed, tested, and approved for production use, update appropriate policies and procedures (for example, production schedules or support).

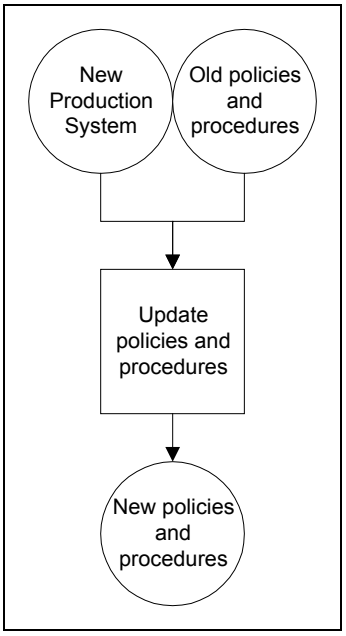


Figure 10–10. Update Policies and Procedures

Activity 5.10: Review (Component 5)

See Figure 10–11. After the system has been installed, tested, and approved for production use, review the status to decide whether to move into the Component 6, Service the Solution.

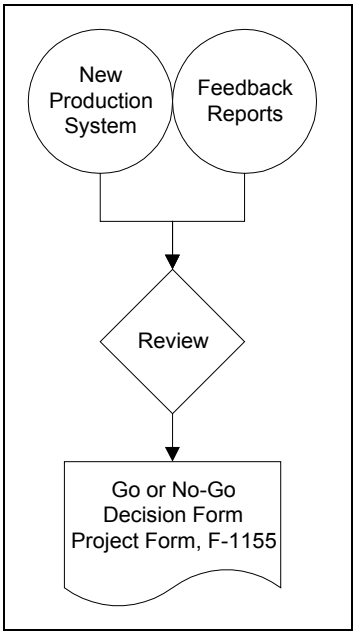


Figure 10–11. Review (Component 5)

11. Component 6. Service the Solution

11.1 Component Overview

The purpose of Component 6, Service the Solution, is to maintain or enhance an existing application or its support environment.

This component is necessarily *very* different from the first five components. The presentation in this chapter is also necessarily very different.

Figure 11–1 provides a high-level illustration of the structure of Component 6. Three different approaches to servicing a solution are shown within the broken line at the bottom of the figure:

- Enhancement
- Release-based maintenance
- Emergency maintenance

Those three approaches are supported by two illustrated elements of configuration management:

- Change management
- Release management

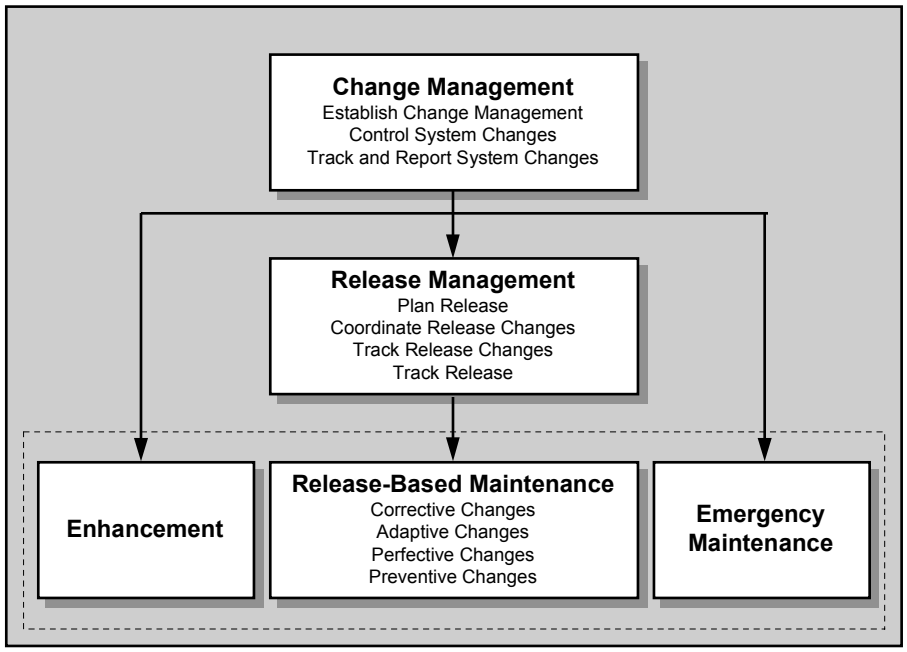


Figure 11–1. High-Level View of Servicing the Solution

Change management evaluates Change Proposals (which may be contractual requests for application system modifications) and Problem Reports and routes them appropriately. Emergency changes are routed to emergency maintenance. Routine changes are allocated to releases and proceed to release planning. A major change is treated as an enhancement.

Release management takes a defined release, plans the release, and manages changes to the release during its development. One result of planning the release is its division into work packages for assignment to different teams.

Enhancement typically requires a major change to some part of an existing application system, perhaps to the system architecture or to the support environment. Often it is a customer who declares a change to be an enhancement. *Within the SDLCM Methodology, an enhancement is treated the same as a new development effort; that is, by beginning with Component 1 and proceeding through the first five components.* This approach provides adequate visibility to the enhancement project and ensures that the system documentation will be updated to reflect the major change. The enhancement process is, therefore, not addressed further in this chapter.

Release-based maintenance is the process of taking a defined work package and defining, executing, and testing the required changes.

Emergency maintenance is an alternative to release-based maintenance for changes requiring immediate action. These types of changes, which are the exception rather than the rule, are followed up with a normal release-based maintenance process.

Figure 11–2 provides a process flow diagram for the top level activities of Component 6 showing the triggers and results. This figure also shows that the output of release-based maintenance flows into integration and then to deployment. Integration is the process of taking the defined work package and integrating it with any other work packages to be deployed. Deployment refers to installing and rolling out the integrated solution. Both are discussed in other chapters of this handbook.

The remainder of this chapter is organized around the major processes of change management, release management, release-based (routine) maintenance, and emergency maintenance. An iconic representation of Figure 11–1 appears at the beginning of each section as a reminder of the relationships among the major processes. For each major process, the following are examined and discussed:

- Activities performed
- Roles that perform those activities
- Products produced
- Critical dependencies
- Configuration management of the products
- Quality assurance

Roles, activities, products, and flow are summarized in:

- Process flow diagrams
- Data flow diagrams
- Activity to work product matrices
- Activity role matrices

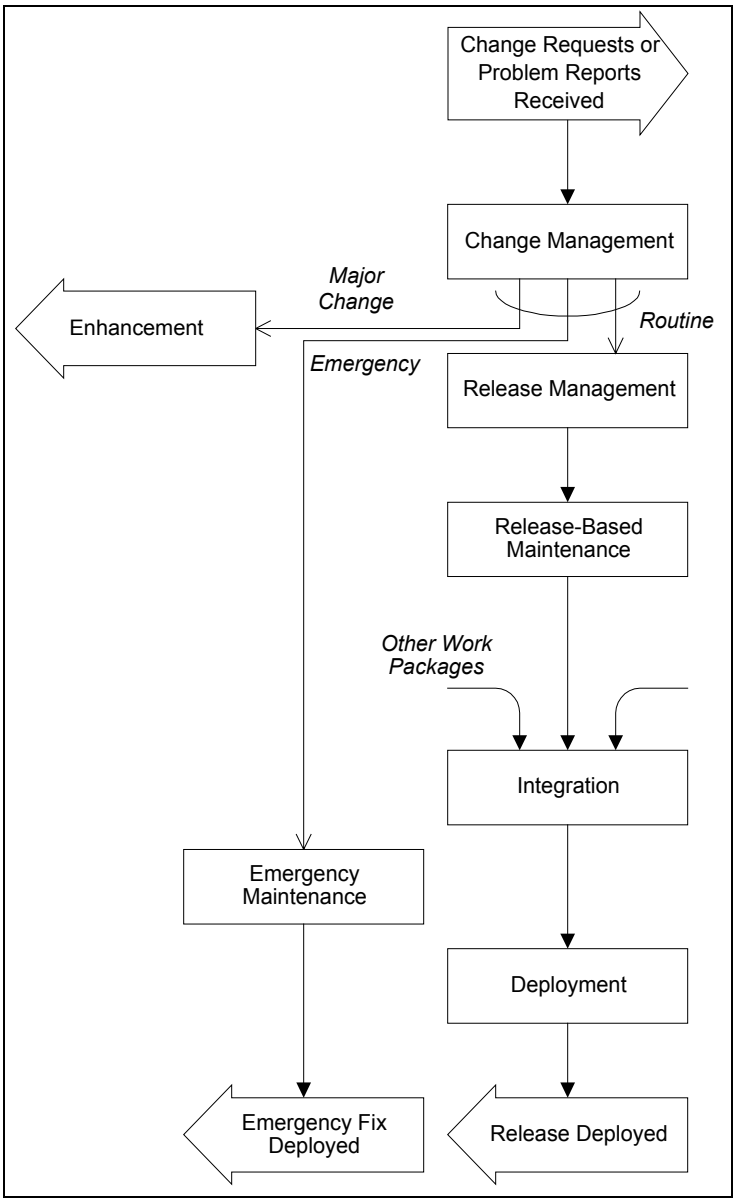
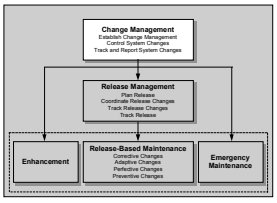


Figure 11–2. Component 6 Process Flow Diagram

11.2 Change Management



This group of activities supports the management of changes to existing systems, including software, hardware, facilities, and staffing. There are three major activities within change management summarized in Figure 11–3:

- Establish Change Management
- Control System Changes
- Track and Report System Changes

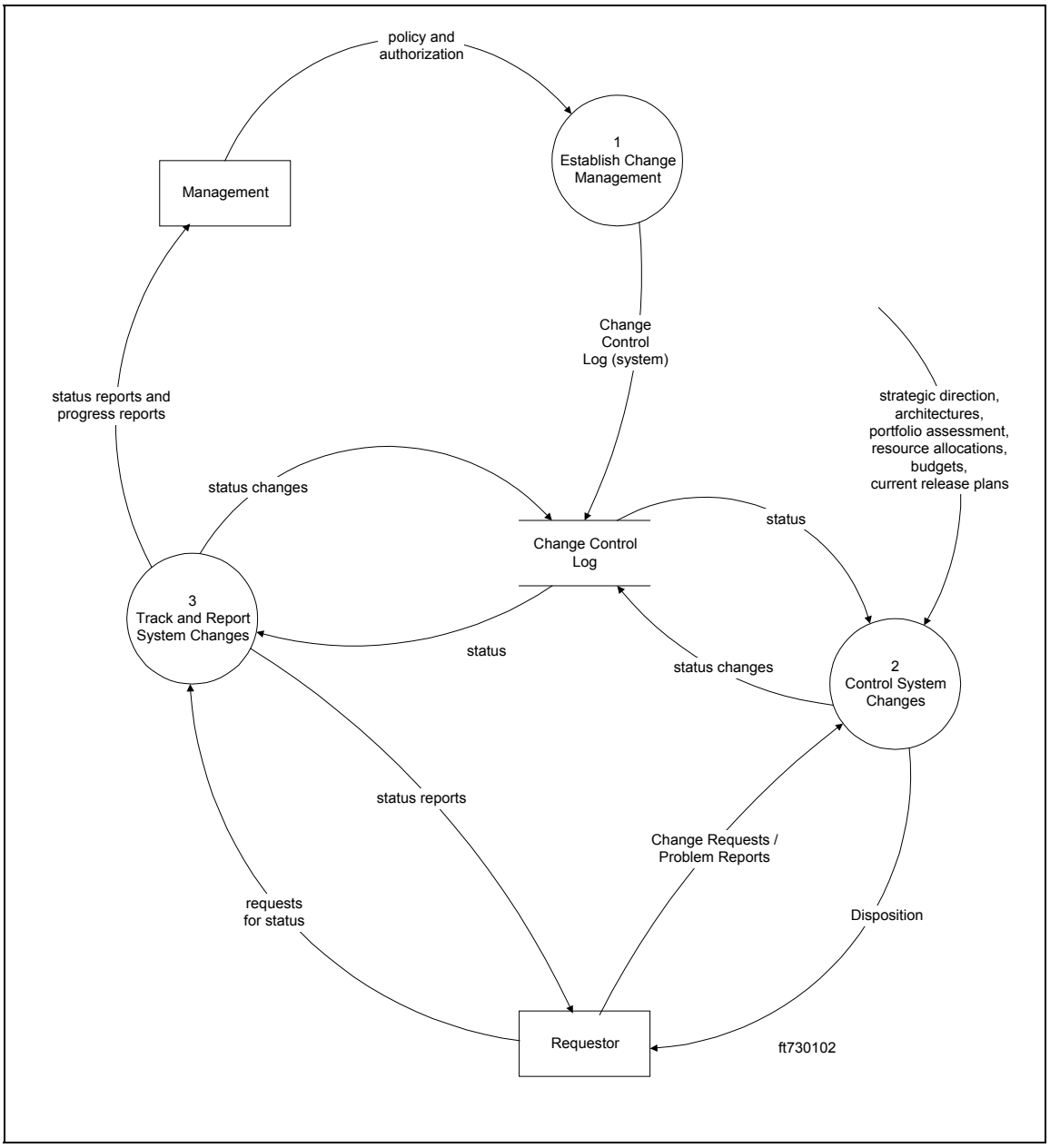


Figure 11–3. Change Management Data Flow Diagram

Roles required for change management include:

- **Configuration Control Board (CCB).** This is any group or individual with authority to review and approve changes. There may be a single CCB or multiple CCBs to handle changes of different scope or magnitude. A CCB may be a single individual. (See the Configuration Management chapter for more information.)
- **Configuration Management Office (CMO).** This is the administrative arm of the CCB. This may be one person or a group.

- **Systems engineering.** This group of architects and analysts on the development or maintenance team evaluates the technical requirements of a change proposal or problem report and estimates the effort, cost, and schedule to implement it.
- **Requester.** This is anyone who requests a change or requests information regarding the status of a change. It may be a user, developer, operator, manager, or anyone else with an interest in the system.
- **Management.** This is meant to include anyone with global interest in the change management process and progress in clearing change proposals and problem reports. It may include the CCB, executive management, business function managers, and others.

The following three subsections discuss the steps involved in the Change Management activities and the roles and products associated with those activities.

11.2.1 Establish Change Management

Establishing change management involves establishing the organizations, roles, responsibilities, processes, policies, procedures, tools, and standards to control changes to existing systems. System changes must be controlled through an orderly process to prevent defects and failures and to maximize business value added.

Change management is an integral function of Configuration Management, which is covered in Chapter 5 of this handbook. For the purposes of this chapter, it is sufficient to know that Configuration Management is responsible for performing the following activities, which play a key role in servicing the solution:

- Acquire and set up the tools to support change management, including an automated Change Control Log.
- Develop a Change Proposal form.
- Develop a change management process covering the requesting, evaluating, approving, allocating, tracking, and reporting of system changes.
- Identify the organizations, roles, and responsibilities for performing the change management process.
- Determine the rules for routing different categories of problems, changes, and issues based on category, size (or effort or cost), and risk.
- Develop standards and procedures for completing change proposals.
- Document the policies, procedures, and standards to implement the change management process.
- Communicate the change management policies, procedures, and standards to the organization.

The outputs of the change management function that help support servicing the solution are:

- Change Control Log (tracking and reporting system)
- Change Proposal form (input document)
- Change Management Policies, Procedures, and Standards

11.2.2 Control System Changes

A report of a problem or a request for a system change may trigger the Control System Change activity as shown in the process flow diagram (see Figure 11–4). The data flow diagram (see Figure 11–5) illustrates the flow of data among the ten lower level activities:

1. **Create and Register Change Proposals**. Initiate the change process by creating and registering Change Proposals. Changes may originate from various sources including users, managers, maintainers, and operators. Changes may originate as a trouble call to a help desk or a completed Change Proposal form.
2. **Research, Validate, and Complete Change Proposals**. Make sure the problem or Change Proposal is well defined, properly classified, and valid; complete the information on the request; and assemble all necessary support information. The important point is to make sure the request is adequately documented without requiring excessive documentation that would slow down the process.
3. **Batch Change Proposals**. Review the Change Proposals to determine whether and how they should be batched into Change Packages for further analysis as a unit. Different reported problems might have the same root cause; they may be symptoms of one underlying problem. Several Change Proposals may be satisfied by one general solution. The idea is to see the larger picture.
4. **Analyze Problems**. Analyze a problem to determine its root cause and to propose potential solutions.
5. **Propose Changes**. Propose the specific technical changes required to implement the Change Proposals. By the end of this activity, you should have described the proposed changes in system terms, but you may not have identified all the various components affected by the change.
6. **Analyze Impacts**. Analyze the impact of a proposed change. If not done already, translate the change into system terms and identify all the components affected by the change. At this point, it is not necessary to specify all the changes required but to get a sense for the size of the effort required—counts may be sufficient—so the CCB will have a sufficient basis for evaluating the merits of the change. Another round of impact analysis will occur early in development, once the requirements have been determined.
7. **Estimate Effort, Cost, and Schedule**. Estimate the effort, cost, and schedule for the change, now that the scope is understood, in order to give the users sufficient information to approve or withdraw a proposed change, or a CCB sufficient information to accept or reject a change.
8. **Assess Cost and Benefit**. Combine the cost, benefit, and schedule information into a cost-benefit analysis that will enable the CCB to review the change as a business decision.
9. **Prepare Change Packages for CCB Review**. Assemble Change Package documentation and route the changes to the proper CCBs for review and action. The criteria used to route Change Packages to different groups should include size, scope, and urgency.
10. **Review and Allocate Change Packages**. CCBs review the Change Packages for merit and priority and then allocate them to particular releases. In doing this, the CCB may remove individual changes from a Change Package and put them in another Change Package, reject them, or defer them indefinitely without assignment to a release.

Likewise, the CCB may reject entire Change Packages or defer them indefinitely without assigning them to a release.

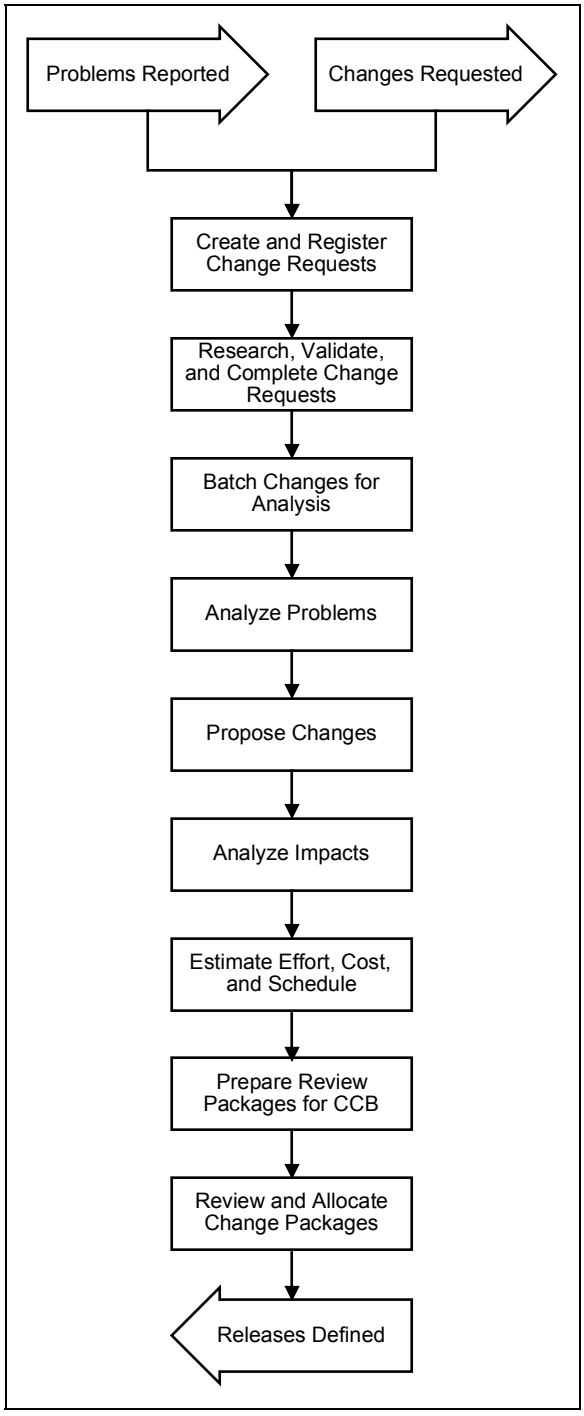


Figure 11–4. Control System Changes, Process Flow Diagram

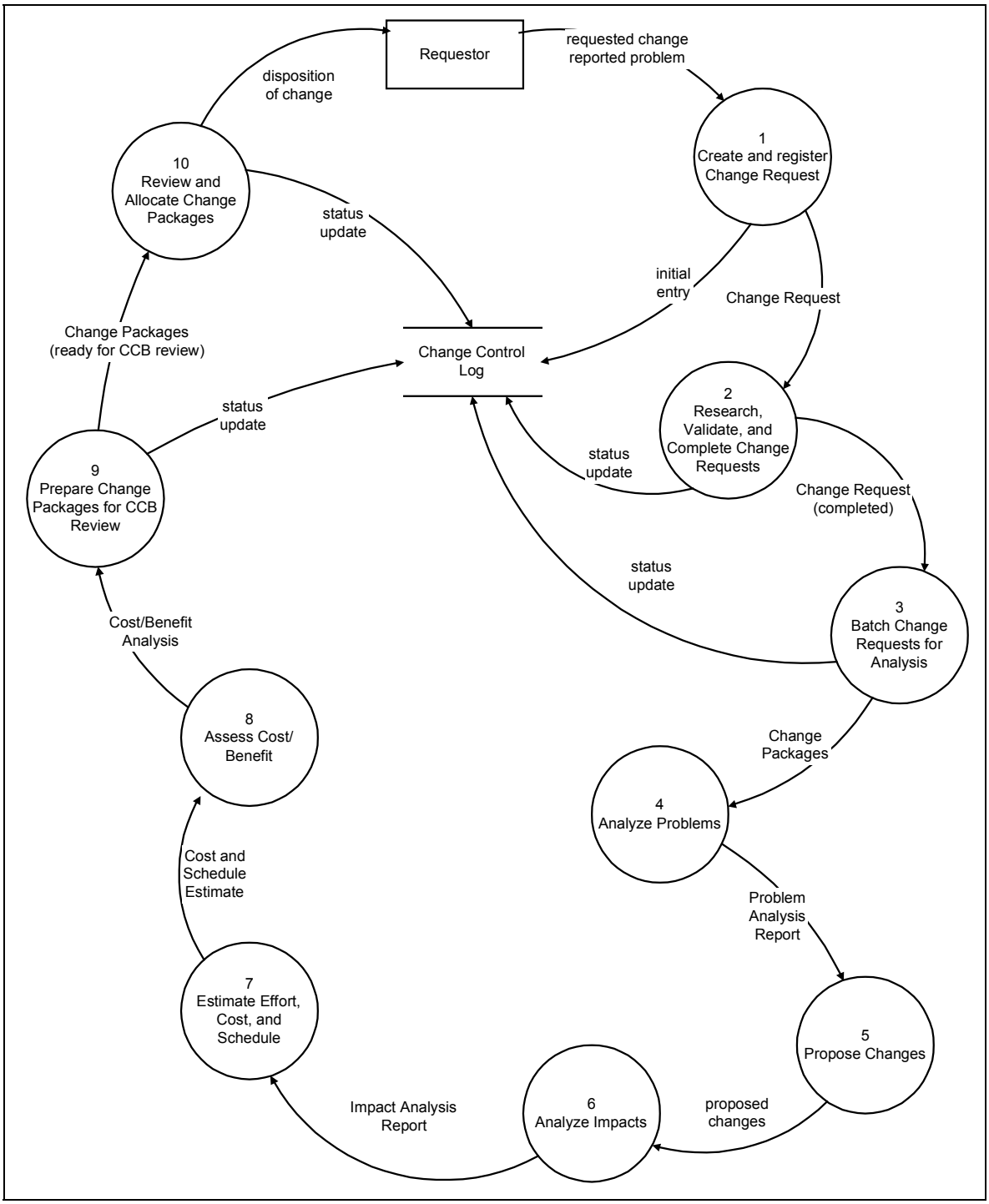


Figure 11–5. Control System Changes, Data Flow Diagram

Table 11–2 associates the ten activities with the roles responsible for their execution.

Table 11–2. Control System Changes, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Create and register change proposals.		R	P						
2. Research, validate, and complete change proposals.			P						
3. Batch change proposals.			P						
4. Analyze problems.			P						
5. Propose changes.		R	P						
6. Analyze impacts.			P						
7. Estimate effort, cost, and schedule.		P							
8. Assess cost and benefit.		P							
9. Prepare change packages for CCB review.		P							
10. Review and allocate change packages.		P						R	

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

11.2.3 Track and Report System Changes

This activity involves tracking and reporting changes in the status of Change Proposals and Problem Reports and assessing the overall progress in addressing system changes. Triggers and results are shown in the process flow diagram (see Figure 11–6). The data flow diagram (see Figure 11–7) illustrates the flow of data among the three lower level activities:

1. **Track Change Status**. Monitor Change Proposal activity to identify changes in Change Proposal status (either directly as an individual Change Proposal or indirectly through association with a change package or release). Update the Change Log from which the status of individual changes can be obtained and the trends can be analyzed.
2. **Report Change Status**. Report the status of individual changes upon request.
3. **Analyze and Report Change Progress**. Summarize management information from the Change Control Log. Analyze and report trends in the process of addressing Change Proposals. This will show whether the backlog is growing or shrinking and various statistics like mean time to failure, mean time to repair, and so on.

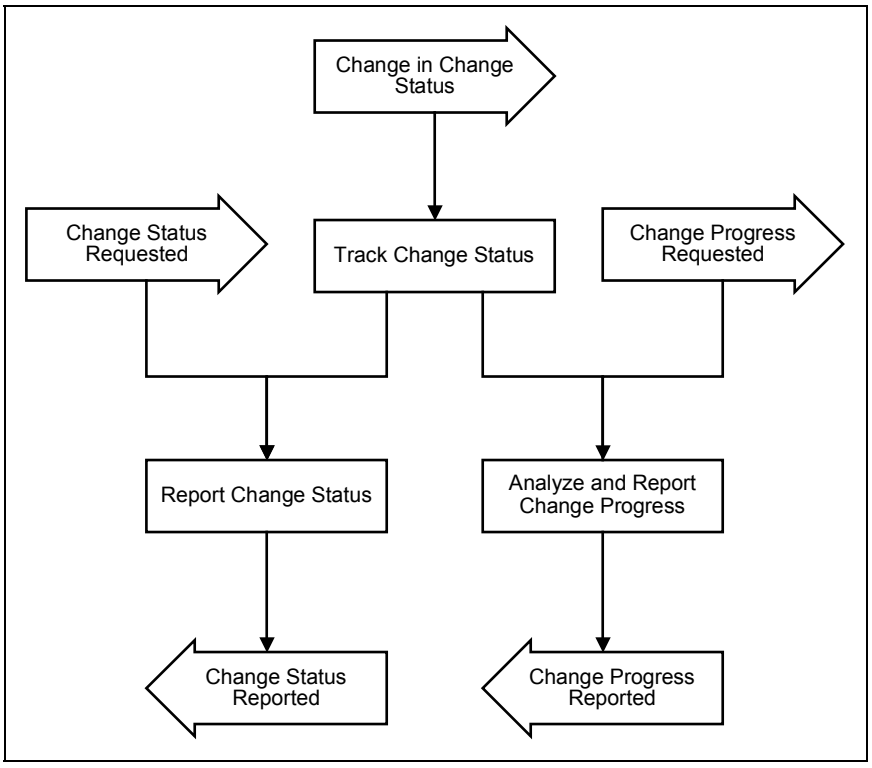


Figure 11–6. Track and Report System Changes, Process Flow Diagram

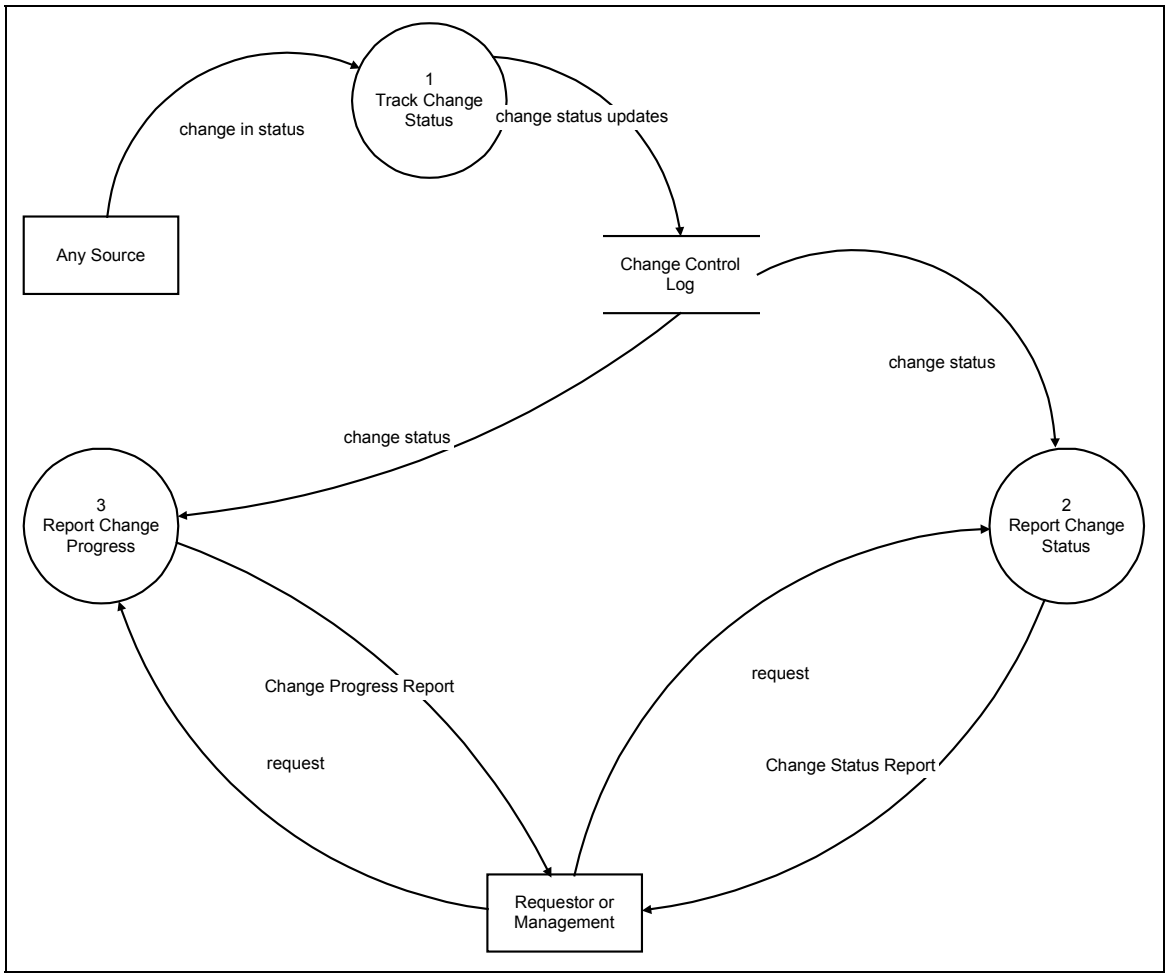


Figure 11–7. Track and Report System Changes, Data Flow Diagram

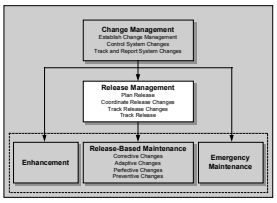
Table 11–3 associates the three activities with the roles responsible for their execution.

Table 11–3. Track and Report System Changes, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Track change status.		P							
2. Report change status.		P							
3. Analyze and report change progress.		P						R	

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

11.3 Release Management



The purpose of this group of activities is to manage a release from the time it has been defined through its deployment. It does not include the initial allocation of change packages to a release. Release Management includes four major activities:

- Plan Release
- Coordinate Release Changes
- Track Release Changes
- Track Releases

The following four subsections discuss the steps involved in the Release Management activities and the roles and products associated with those activities.

11.3.1 Plan Release

The purpose of this activity is to plan the release from an engineering point of view, as opposed to a project management point of view. It verifies the means for development coordination and establishes the structures for managing the components being created or modified by the release.

A defined release triggers this activity as shown in the process flow diagram (see Figure 11–8). There are seven lower level activities:

1. **Verify Development Coordination**. Verify that a development coordination capability exists to coordinate all teams involved in the release and to coordinate the release activities with the larger enterprise. This activity ensures that the organizations, responsibilities, capabilities, working relationships, policies, procedures, and infrastructure exists to provide adequate coordination.
2. **Establish Configuration Management**. Verify and if necessary establish configuration management for the release. Configuration management procedures, tools, and standards should already exist for an application system that is under maintenance, but they should be confirmed. Other configuration management planning is required for each new release.
3. **Review or Establish Standards**. Review all applicable standards to be sure they are adequate for the project and to create additional standards if needed. Include standards for both products and management documents. Confirm how the standards will be enforced.
4. **Review or Establish Common Services**. The purpose of this activity is to plan for reuse or sharing of components. Review existing reuse policies or establish policies on reuse. Establish what common services are included within the scope of the release, what new common services are required, and what new common services might be contributed from existing components within the scope of the release.
5. **Plan Release Development and Define Work Packages**. Plan how the release should be developed. Divide the release into loosely coupled work packages that can be worked on by separate teams. Scope the work packages to minimize conflicts and coordination requirements among the teams. Changes that were previously allocated to releases are now allocated to work packages within the release. Any given configuration item is generally assigned to only one work package.

6. **Plan Release Integration**. The purpose of this activity is to plan the integration, integration testing, acceptance, and data conversion testing (if applicable) for the release. This includes any planning for the use of pilots.
7. **Plan Release Deployment**. Plan the deployment of the accepted release to all of the various deployment sites and plan for post-deployment support and operational testing.

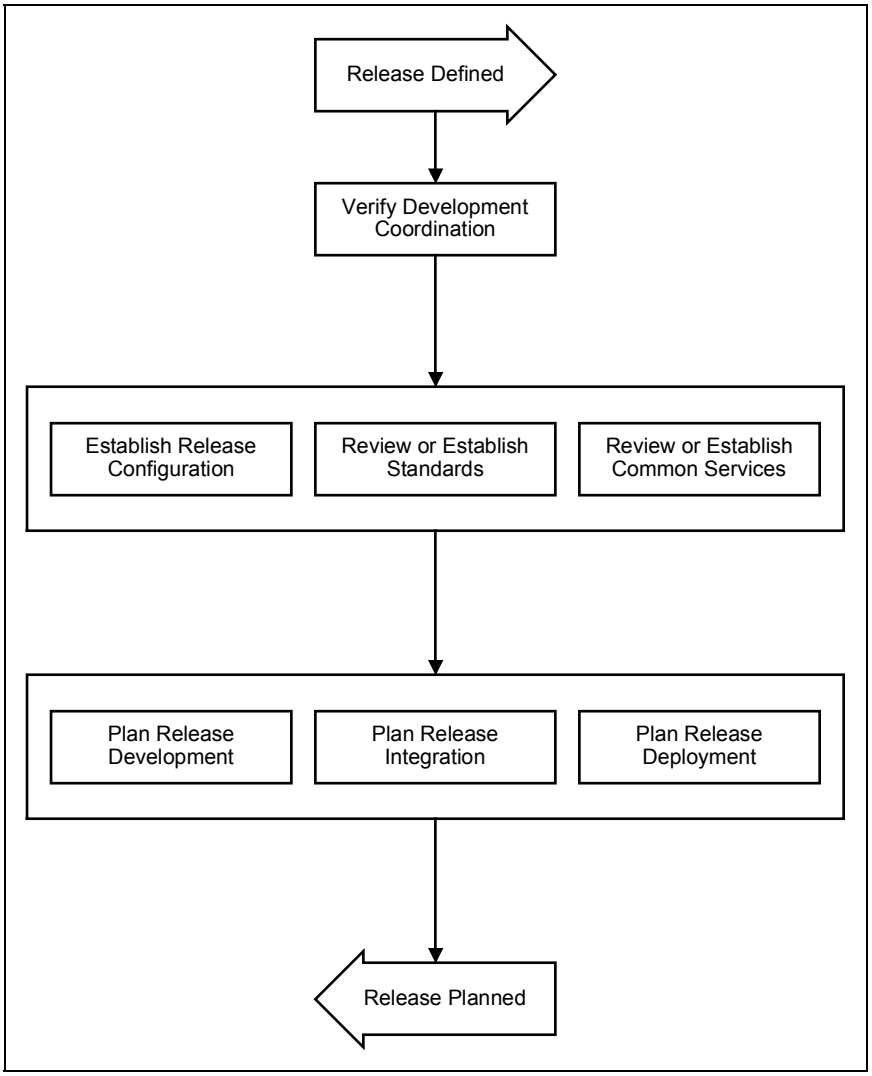


Figure 11–8. Plan Release, Process Flow Diagram

Table 11–4 associates the ten activities with the roles responsible for their execution.

Table 11–4. Plan Release, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Verify development coordination.	P	S							
2. Establish configuration management.								P	
3. Review or establish standards.		P							
4. Review or establish common services.		P							
5. Plan release development and define work packages.		P							
6. Plan release integration.		P							
7. Plan release deployment.	P	S							

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

11.3.2 Coordinate Release Changes

The purpose of this activity is to maintain the release plans and to coordinate changes within the release among teams and with enterprise models, standards, plans, and such. Changes are of two types: release plan changes and engineering changes. Release plan changes are changes to release scope, schedule, or cost. Engineering changes are changes to common models, standards, common services, interfaces, common databases, or common infrastructure. Engineering changes may be proposed for use within the release (such as what version of a particular product will be used) or may be proposed by the release for enterprise-wide use, such as a suggested change to the corporate data model.

This activity is triggered when a change proposal is routed to routine released-based maintenance. (See Figure 11–9.) There are nine lower level activities:

1. **Create and Register Change Proposal**. The purpose of this activity is to formalize a requested change to enable its analysis and resolution and to log the change to permit its tracking and prevent its being forgotten.
2. **Analyze Impacts**. The purpose of this activity is to analyze the impact of a requested change. What would have to change to implement the request? How does the change affect other teams working on the release? How does it affect release cost, schedule, or product quality? Does it require a change at the enterprise level?
3. **Negotiate Technical Changes**. The purpose of this activity is to negotiate changes where there is a difference of opinion, for example, if a project wants a data model change that corporate data administration is fighting.
4. **Estimate Effort, Cost, and Schedule**. The purpose of this activity is to estimate the effort, cost, and schedule required for implementing the change.

5. **Reassess Cost and Benefit**. The purpose of this activity is to assess the impact that the change would have on the cost and benefit measures for the release. This is intended to provide enough information for the CCB to make a business decision on the proposed change. This activity is performed only if there is an impact on release cost, schedule, or content. Many engineering changes will not affect cost, schedule, or content and thus will not require CCB review.
6. **Package Requests for CCB Review**. The purpose of this activity is to prepare the Change Proposals for presentation to the CCB. Present the facts and required decisions clearly and concisely. Present the questions in order of their importance.
7. **Negotiate and Approve Release Changes**. The purpose of this activity is to negotiate changes to the release plans. Such changes might include a slippage of the date to preserve content or a removal of content to preserve the date.
8. **Implement Changes to Plans**. The purpose of this activity is to implement the agreed-upon changes to the various plans, including plans at the enterprise, release, and project levels. The result should be a consistent set of plans.
9. **Implement Engineering Changes**. The purpose of this activity is to implement the agreed-upon changes to the various common engineering documents and components. These include changes to models, standards, common services, interfaces, databases, and infrastructure at the individual, team, project, release, and enterprise levels. The results should be a consistent set of engineering documents and shared components.

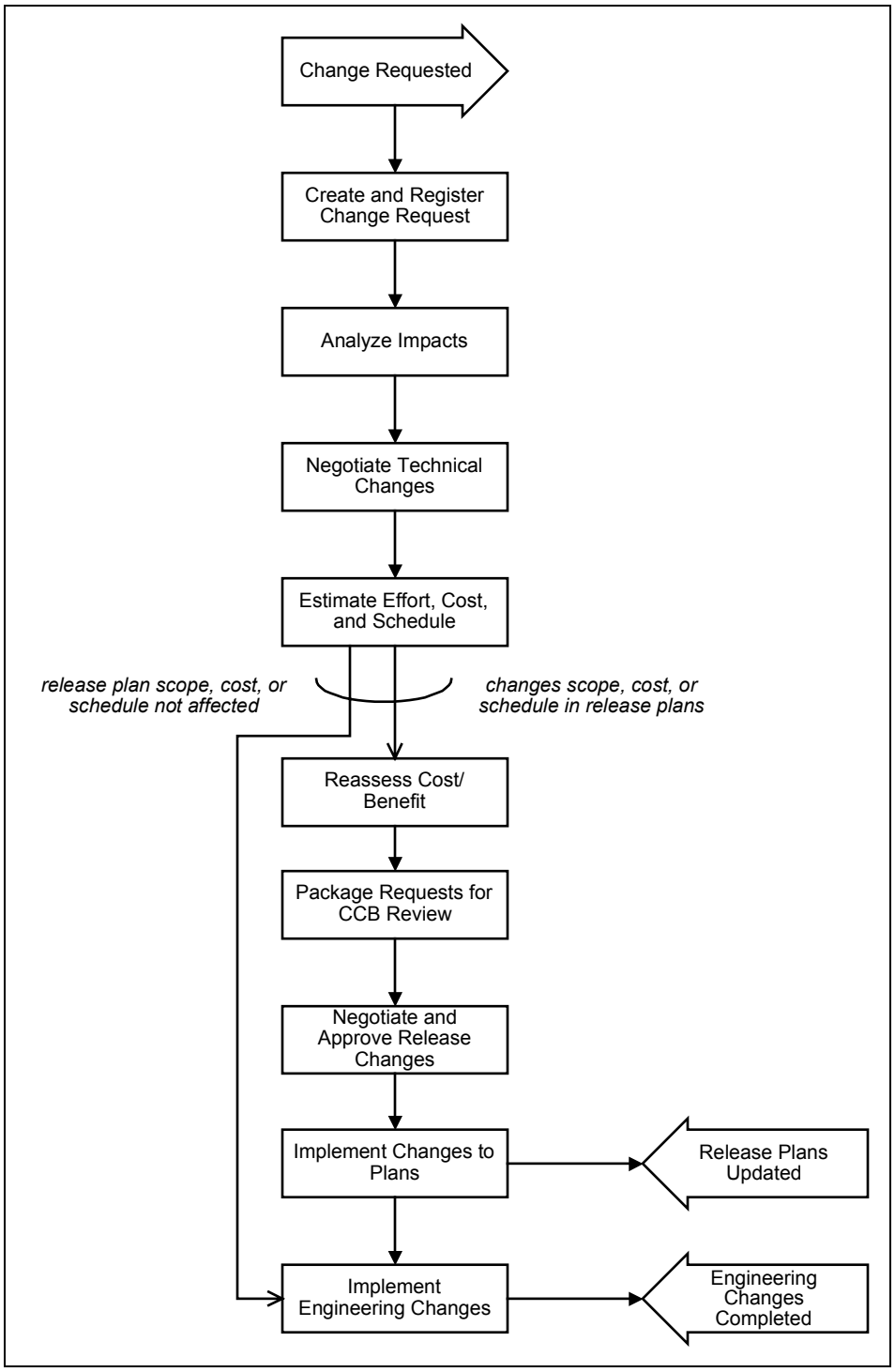


Figure 11–9. Coordinate Release Changes, Process Flow Diagram

Table 11–5 associates the ten activities with the roles responsible for their execution.

Table 11–5. Coordinate Release Changes, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Create and register change proposal.		P							
2. Analyze impacts.		R	P						
3. Negotiate technical changes.	P	P	P						
4. Estimate effort, cost, and schedule.		P	S						
5. Reassess cost and benefit.	P	P							
6. Package requests for CCB review.		P	S					R	
7. Negotiate and approve release changes.		P						P	
8. Implement changes to plans.		P							
9. Implement engineering changes.			P						

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

11.3.3 Track Release Changes

The purpose of this activity is to track and report changes in the status or change proposals and to assess the overall progress in addressing change proposals.

This activity is triggered when a change proposal changes status or there is a request to report status or progress of changes. (See Figure 11–10.) There are three lower level activities:

1. **Track Change Status**. The purpose of this activity is to monitor change proposal activity to identify changes in change proposal status (either directly as an individual Change Proposal or indirectly through association with a change package or release). This updates the Change Log from which the status of individual changes can be obtained and trends analyzed. The Change Report and Change Log for these changes can be the same as used for changes requested for existing systems.
2. **Report Change Status**. The purpose of this activity is to report the status of individual changes upon request.
3. **Analyze and Report Change Progress**. The purpose of this activity is to analyze and report trends in the process of addressing change proposals. This will show whether the backlog is growing or shrinking, among other things.

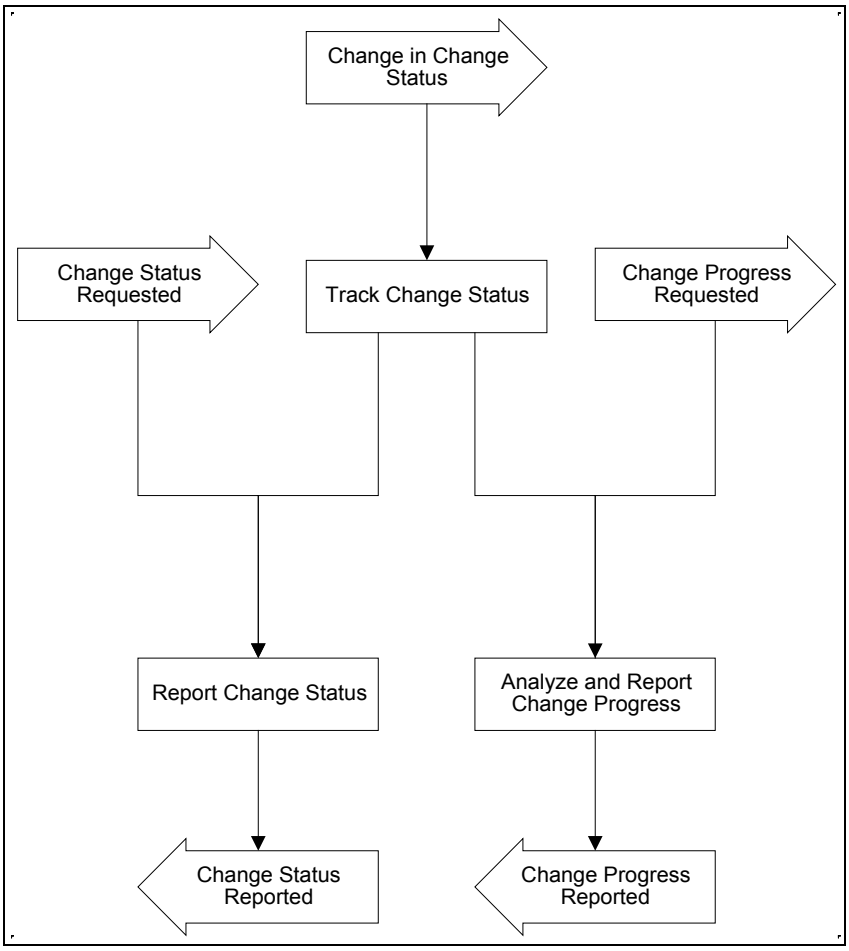


Figure 11–10. Track Release Changes, Process Flow Diagram

Table 11–6 associates the ten activities with the roles responsible for their execution.

Table 11–6. Track Release Changes, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Track change status		P							
2. Report change status		P	S						
3. Analyze and report change progress		P						R	

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

11.3.4 Track Releases

The purpose of this activity is to track the status and progress of releases. It is fed by the project status reporting from the project management processes. A project can span several releases, and a release can span several projects. The primary information monitored is the estimated completion date and the estimated cost at completion for the release. This is needed to be able to report the expected deployment dates for changes being implemented by the release. (There may also be intermediate milestones defined for the release, but these are more important for project management and program assurance than for change management.)

This activity is triggered when a release changes status or there is a request to report status or progress of changes. (See Figure 11–11.) There are three lower level activities:

1. **Track Release Status**. The purpose of this activity is to monitor the planned cost, schedule, and content of the release as it changes based on agreed-upon requested changes and actual events. At any point in time, you should be able to identify the projected completion date, the projected cost, and the planned content of the release (that is, which change packages are included).
2. **Report Release Status**. The purpose of this activity is to report, upon request or regular schedule, the projected completion date, the projected cost, and the planned content of the release.
3. **Analyze and Report Release Progress**. The purpose of this activity is to report, upon request or regular schedule, any information showing rates of progress. This might be stated in terms of earned value if this approach has been implemented.

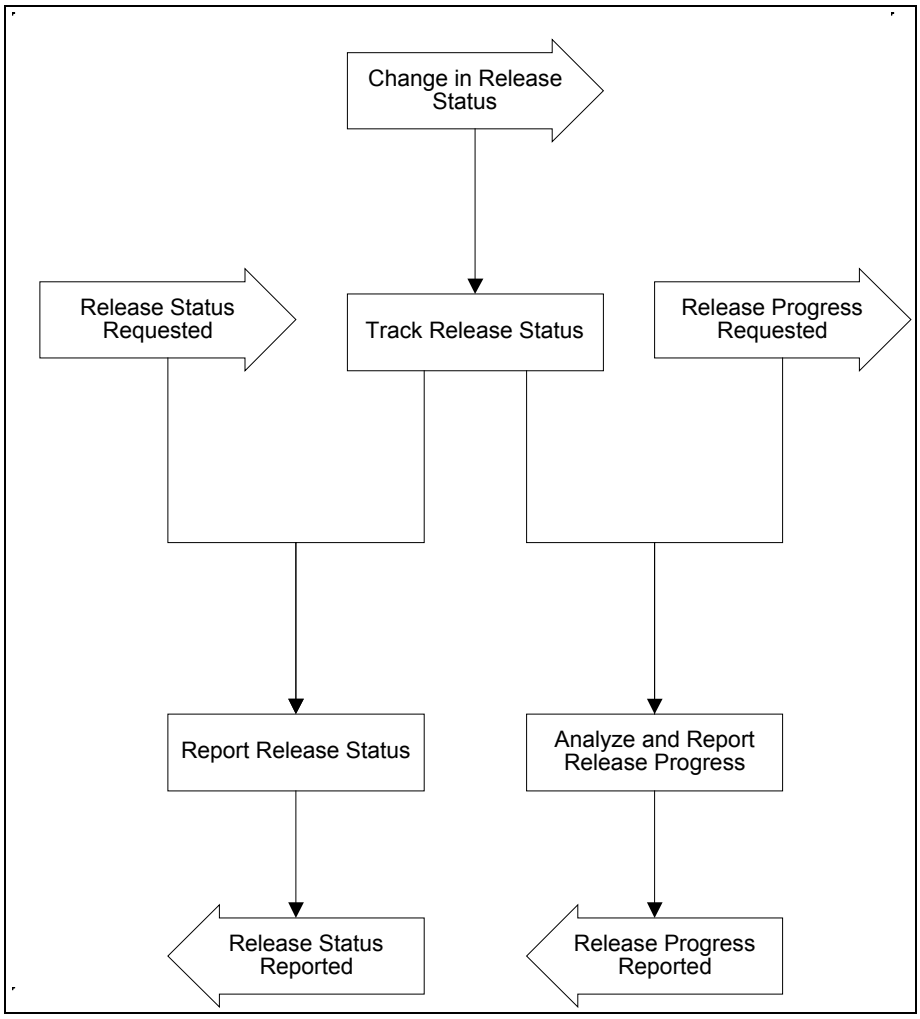


Figure 11–11. Track Releases, Process Flow Diagram

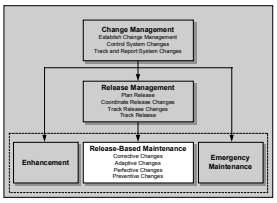
Table 11–7 associates the activities with the roles responsible for their execution.

Table 11–7. Track Releases, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Track release status.		P							
2. Report release status.		P	S						
3. Analyze and report release progress.		P						R	

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

11.4 Release-Based Maintenance



The Routine Maintenance process addresses maintaining and evolving existing systems. It focuses on making changes to an existing environment, taking the operational system and its infrastructure as constraints: the changes have to be built on the underlying structure and have to be compatible with the existing system architecture.

There are four types of changes:

- Corrective changes
- Adaptive changes
- Perfective changes
- Preventive changes

Routine Maintenance may be performed continually and in parallel with other development paths. Triggers and results are shown in the process flow diagram (see Figure 11–12). The data flow diagrams (see Figure 11–13 and Figure 11–14) illustrates the flow of data among the eighteen lower level activities:

1. **Determine Type of Change**. Understand the reason for the change in order to determine the type of change. The type of change information helps you to select the appropriate sequence of activities and their level of detail. Do this for each change contained in the Work Package. This activity starts the Release-Based Maintenance Process and is usually performed after a Work Package has been received. This activity will be completed when each change has been classified as *corrective* or *non-corrective* (that is, adaptive, perfective, preventive). Group changes into sets of related changes.
2. **Extend Requirement Analysis**. Extend the Requirements Analysis to understand the need that caused the Change Proposal. Refine to a more detailed level to produce input for subsequent design activities.
3. **Extend Problem Analysis**. Refine the initial Problem Analysis that was done after receiving the Problem Report until you can identify the underlying cause. Keep in mind that the results of this activity will be the basis for designing the change.
4. **Extend Existing System Analysis**. Create, refine, or update the model views or equivalent system documentation needed for System Evolution and Maintenance. Keep in mind that the results of this activity will be needed for designing the change. This activity may be invoked and performed in parallel to ongoing work when additional, new, or more detailed information is needed. This may happen as a result of a change in system functionality or configuration or during Problem or Requirement Analysis.
5. **Design Change at High-Level**. Work out a high-level design for implementing the change. Focus on the logical level of the programs, the interaction among them or the data entities if the changes are in that area. The results of this activity will be the basis for the implementation of the change. Try to keep the design compatible and aligned with the existing architecture and operation.
6. **Review High-Level Design**. Review the High-level Design for the change. Validate that the design basically adheres to the current architecture and its use and that necessary

changes are compatible to them. Approve or update related estimates for resources, schedule, and costs.

7. **Reassess Impact**. Reassess the impact of the change in the context of the current system and the modifications to be done. If necessary, refine the Impact Analysis in order to get to a valid and agreed assessment of the consequences of the change.

   This activity incrementally creates, updates, or refines information on the change's impact upon the existing system architecture. It may be iterated. As Analysis proceeds or resumes, its scope may evolve from deeper to higher level of detail as well as from rather limited to broader scope.

8. **Decide Whether to Proceed**. Decide whether to promote the change to the implementation activity in the proposed way or to turn the change back for reconsideration. At this time, the impact of the change, the resources needed to implement it, and both the schedule and costs estimates have been validated; and they describe the different aspects of the change correctly.

   In the case the change is turned back, there are two choices:

   (1) Create a different design by repeating the prior steps beginning with "Design Change at High-Level" or

   (2) Raise an issue and send the change to back to the Change Management process.

9. **Implement Change**. The following paragraphs describe the activity "Implement Change" in a generalized way that can be tailored to different development approaches.

   Implementation is a composite activity that can vary greatly in size and complexity given a particular problem set, environment, and development approach. However, regardless of the specific development approach chosen, implementation is made of three kinds of basic activities: design, construction, and validation (or design, code, test). The basic implementation approaches are constructed by wiring together these activities in different sequences and iterations, for different units of scope and time, in different technical environments, and with different human dynamics:

   One option is an iterative learning approach, designing the business process change and then deriving the system change, going through several cycles of design, code, and validate, that allow for learning during the development process and uncovering hidden needs.

   Another option is an accelerated approach, going through several tightly scheduled time boxes consisting of design, code, validate in a lab environment, with functionality made secondary to schedule.

   Another option is an incremental waterfall approach, with larger chunks of design, code, and validate, but with greater emphasis of formal validation of requirements and design.

   Another option is a package-based approach, with requirements transmuting into selection criteria and selection becoming part of the design activity (as in adding a COTS module onto an existing application).

   If prototyping is used, go through several iterations of design, construction, and validation of a prototype as a means to discover hidden requirements and to arrive at a design for the target system.

After the prototyping stage is complete, the coding stage completes the armor-plating process if a limited function prototype is being evolved into the target system; but if only simulation prototyping was used, the coding stages is the construction of the target system in the target environment.

If a package-based approach is selected, understand Design as to include package selection as well.

If a requirements-driven approach is selected, create the Design by use of the Requirement Analysis Technique and iteration to more detailed level.

To assemble an implementation approach, take the basic building blocks, apply tailoring to the specific context, scope, and level of detail, and iterate as needed to cover the whole change. This will provide an implementation path that is tailored to the change and the constraints of the existing environment and satisfies the preferences of the customer.

Start each path variant with an enabling activity that establishes the prerequisites in order to perform the implementation activities.

There are four steps for implementing a change:

- Prepare for Implementation. Establish or validate the prerequisites that are needed for implementing the change. Include both technical and organizational aspects to be able to continue work with designing the next level of detail of the change.
- Design Change detailed. Refine the High-Level Design of the change down to the Unit or Module level. The goal of this activity is to develop a physical design for programs and the referenced data so that coding can start. Several iterations may be necessary until this level of detail will be reached.
- Code the Change. Code the change on the basis of the detailed design that has been created before. In parallel to this work, create Test Cases for testing code sections during implementation and later for Unit Testing.
- Short-Test Code (optional). Perform some short tests on selected code sections to get confirmation that the changed modules will basically work and are mature for entering Unit Testing. Short Testing may help to detect obvious implementation or unit design errors very early. The test takes place in the Personal Library. Proceed with coding on the next change item after successful Short test.

10. **Unit Test Change**. When the Implementation has been completed, test the new or changed functions to make sure that they satisfy the requirements and needs described in the Change Proposal or Problem Report. Within Unit Test, focus on the functionality of the change.
11. **Consolidate Changes of Work Package**. Integrate all modules related to the changes of the Work Package with the unchanged modules of the application to be able to perform the Work Package Integration Test.
12. **Perform Work Package Integration Test**. After having integrated all changes of a Work Package, test the new or modified functions of the entire Change Package. Within this test, focus on the communication and interaction between the changes of the Work Package when using the these functions.

13. **Perform Regression Test on Application**. Perform Regression Testing on Application level to ensure that functionality that should persist has not been affected by the new or modified functions introduced with the Change Package.
14. **Integrate Release**. Integrate all Work Packages of the Release with the remaining unchanged modules of the Release. This activity links together the non-modified components with the new or changes module for the first time. It prepares Integration and Regression Testing for the entire Release.
15. **Test the Release**. After the release has been integrated, this activity determines if the existing parts of the release harmonize with new functionality introduced by the Work Packages. Within this test, focus on the interaction of the subsystems or major components of the system or release when using the new functions.
16. **Perform Regression Test on Release**. Perform Regression Testing on Release level to ensure that the existing functionality of the Release has not been affected by the new or modified functions introduced by the Change Packages.
17. **Plan Testing (ongoing).** Test Planning is an ongoing activity in which new information is added to existing documents as it arises. It proceeds in parallel with the design and implementation process, creating Test Cases as soon as they can be defined. Follow this approach if it is feasible.
18. **Document Changes (ongoing).** Document the change on its way through the development life cycle. This documentation is needed primarily for understanding the change. It also helps to keep the system information up to date and correct.

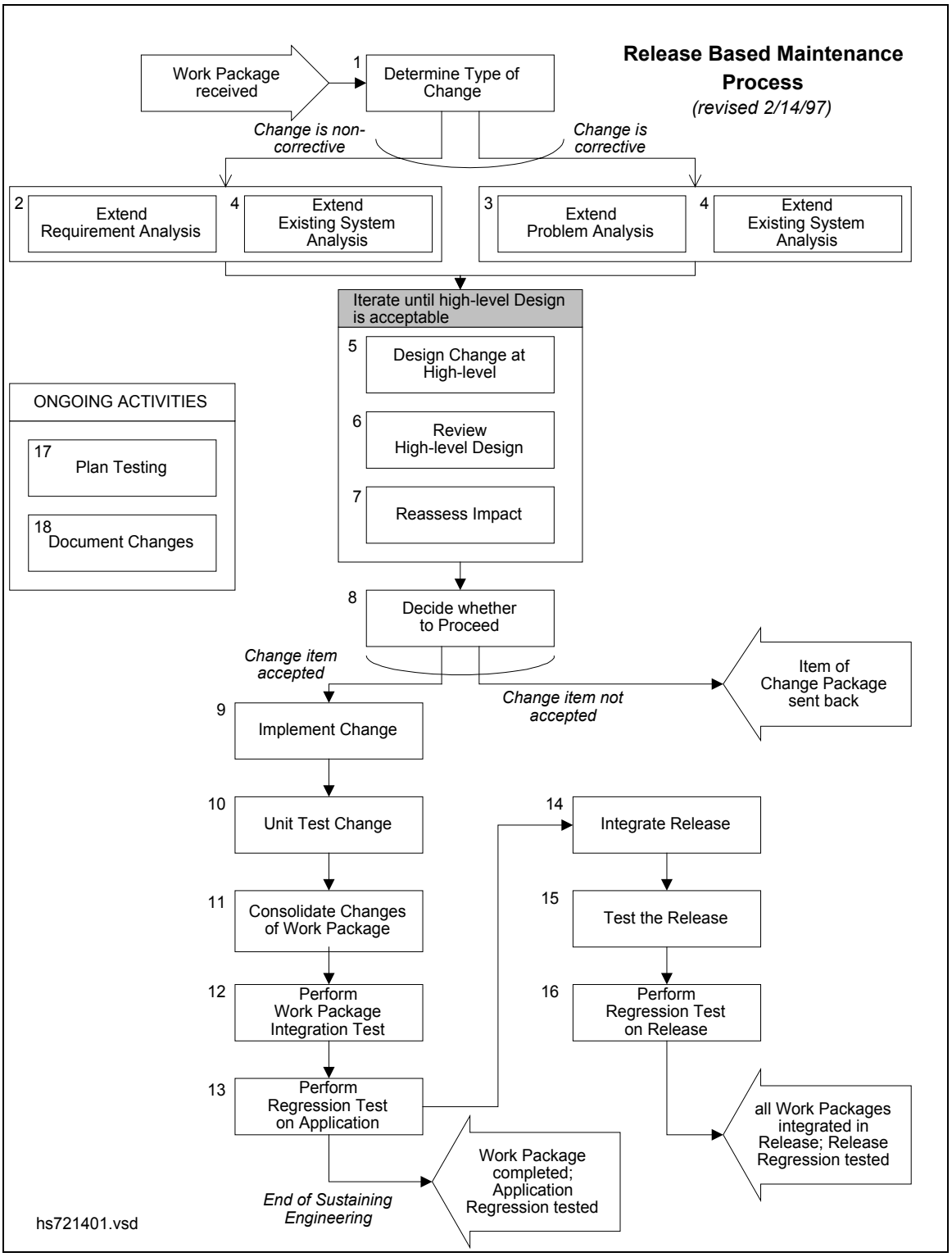


Figure 11–12. Release-Based Maintenance, Process Flow Diagram

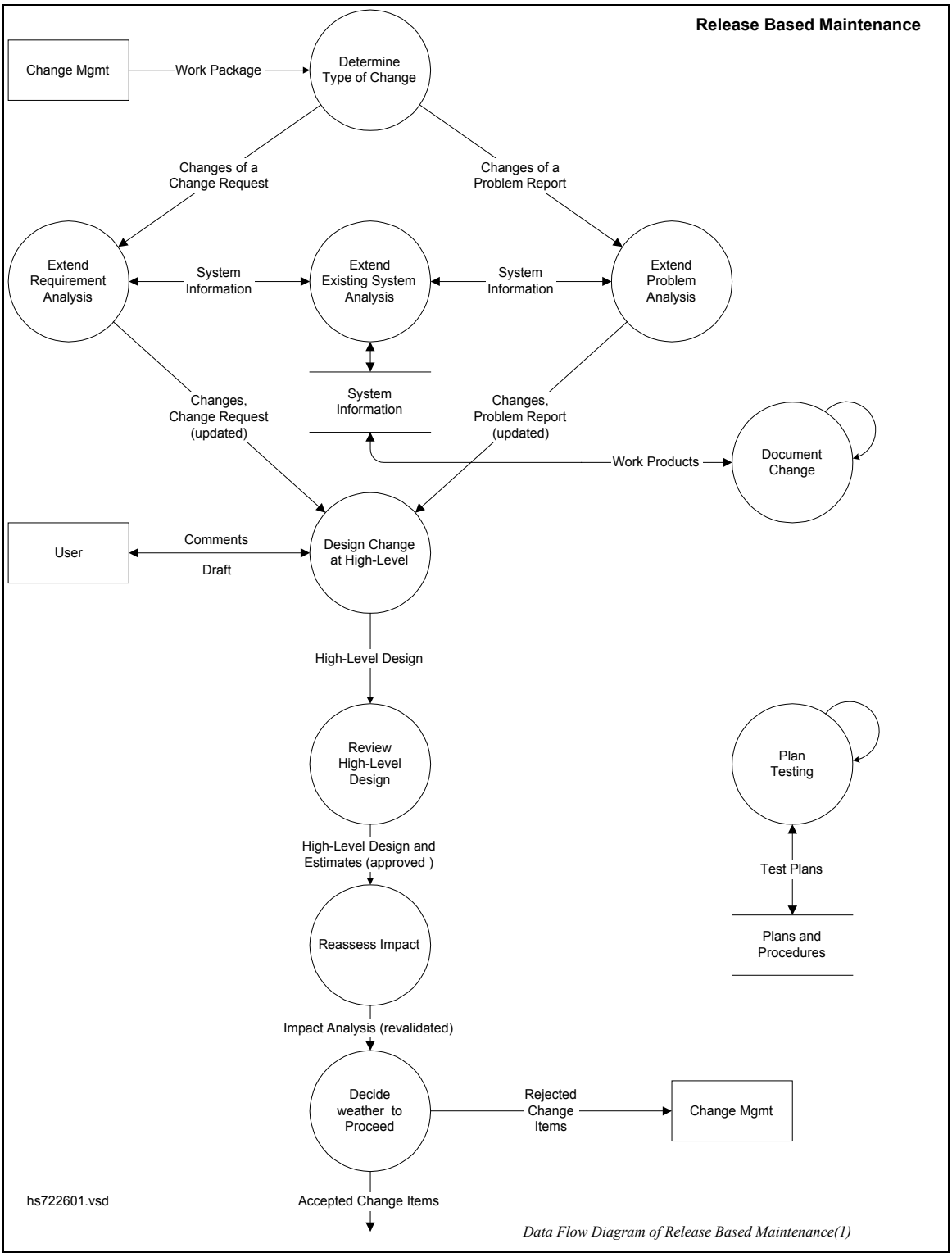


Figure 11–13. Release-Based Maintenance, Data Flow Diagram (1 of 2)

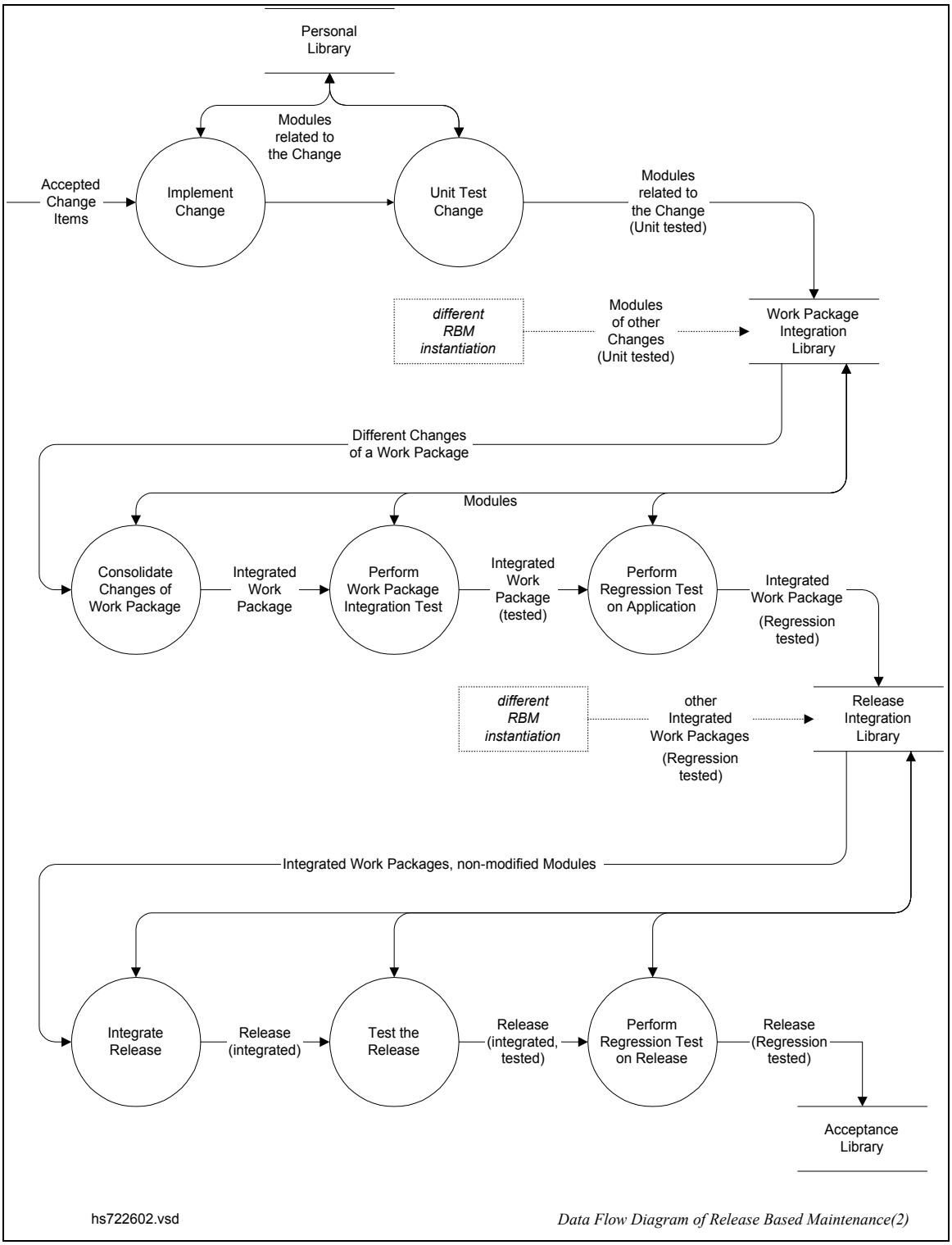


Figure 11–14. Release-Based Maintenance, Data Flow Diagram (2 of 2)

Table 11–8 associates the 18 activities with the roles responsible for their execution.

Table 11–8. Release-Based Maintenance, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Determine type of change.			P						
2. Extend requirement analysis.			P						
3. Extend problem analysis.			P						
4. Extend existing system analysis.			P						
5. Design change at high-level.			P						
6. Review high-level design.		S	P						
7. Reassess impact.		P							
8. Decide whether to proceed.	P								
9. Implement change.			P						
10. Unit test change.			P						
11. Consolidate changes of work package.			P						
12. Perform work package integration test.			P						
13. Perform regression test on application.			P						
14. Integrate release.			P						
15. Test the release.			P						
16. Perform regression test on release.			P						
17. Plan testing (ongoing).			P						
18. Document changes (ongoing).			P						

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

Figure 11–15 depicts the flow of a software module through the various software libraries during the development, integration, testing, and deployment processes. The figure includes the paths for both routine (release-based) maintenance and emergency maintenance (addressed in the next section).

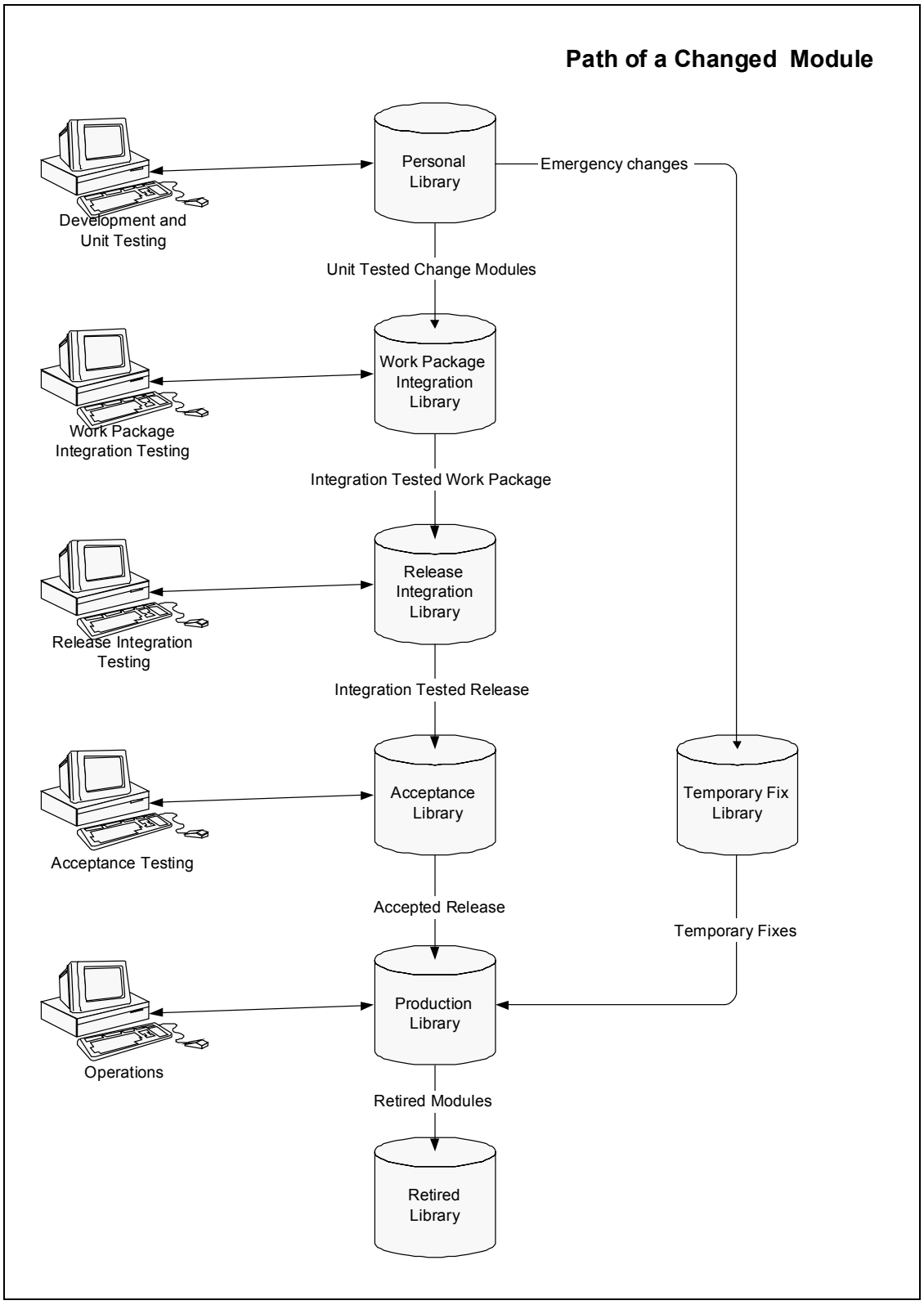
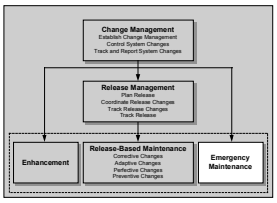


Figure 11–15. Software Libraries for Maintenance

11.5 Emergency Maintenance



Emergency Maintenance addresses maintenance required in emergency situations. An Emergency situation may arise during system operation when system errors with severe impact occur and cannot be handled by the system administration. Such errors must be fixed as quickly as possible to return to operation.

After an emergency situation is fixed by an Emergency Maintenance intervention, the related Problem Report is reconsidered by Change Management and fed into the queue of pending changes to the system. Emergency Maintenance is often a temporary fix to solve an urgent problem very quickly and normally is often not suitable as a lasting change to the system. Thus, it should go into the short list of changes to be included in one of the next releases and then become a permanent and fully quality-assured change to the system.

To ensure consistent decisions with minimal delay, control of the Emergency Maintenance activities is concentrated in an Emergency Management role within Change Management. Emergency Management's primary responsibility is to control the Emergency Maintenance processes and to make the decisions effectively and very fast in an area where system and business issues overlap. It also serves as the interface between Operations and the Emergency Maintenance Team. Emergency Management may consist of only one experienced individual with sound knowledge in both the systems and business worlds. Alternatively, there might be two individuals acting as an efficient team.

It is vital for performing Operations and Emergency Maintenance effectively that information on the system is available and reliable. If such information is not yet available, apply analyze the existing system to establish at least the minimum set of required documentation or refine the existing information if there are gaps before the first emergency situation occurs.

Figure 11–16 gives a high level overview of how Emergency Maintenance fits into the overall cycle of system maintenance and operation. The boxes represent process groups, the arrows the flow of the indicated work products. The figure illustrates the following:

- During Operation, an emergency situation occurs. After a first diagnosis, a Problem Report is sent to Emergency Management.
- Emergency Management assesses the gravity of the problem, different solution approaches, and decides whether an Emergency Maintenance Intervention is needed.
- If it is an instance for Emergency Maintenance, Emergency Management sends the Problem Report to the Emergency Maintenance Team.
- The Emergency Maintenance Team then works to solve the problem by work-arounds or adequate system changes (varies from patches to source code changes).
- After a solution has been identified, implemented, and sufficiently tested, it is fed into production environment.
- Emergency Management (within Change Management) receives the updated Problem Report and documentation on the emergency fix and then decides whether the change will be promoted into the queue of candidate changes for future releases.
- Emergency Management is responsible that Emergency Maintenance interventions are tracked and reported to Operations and Change Management.

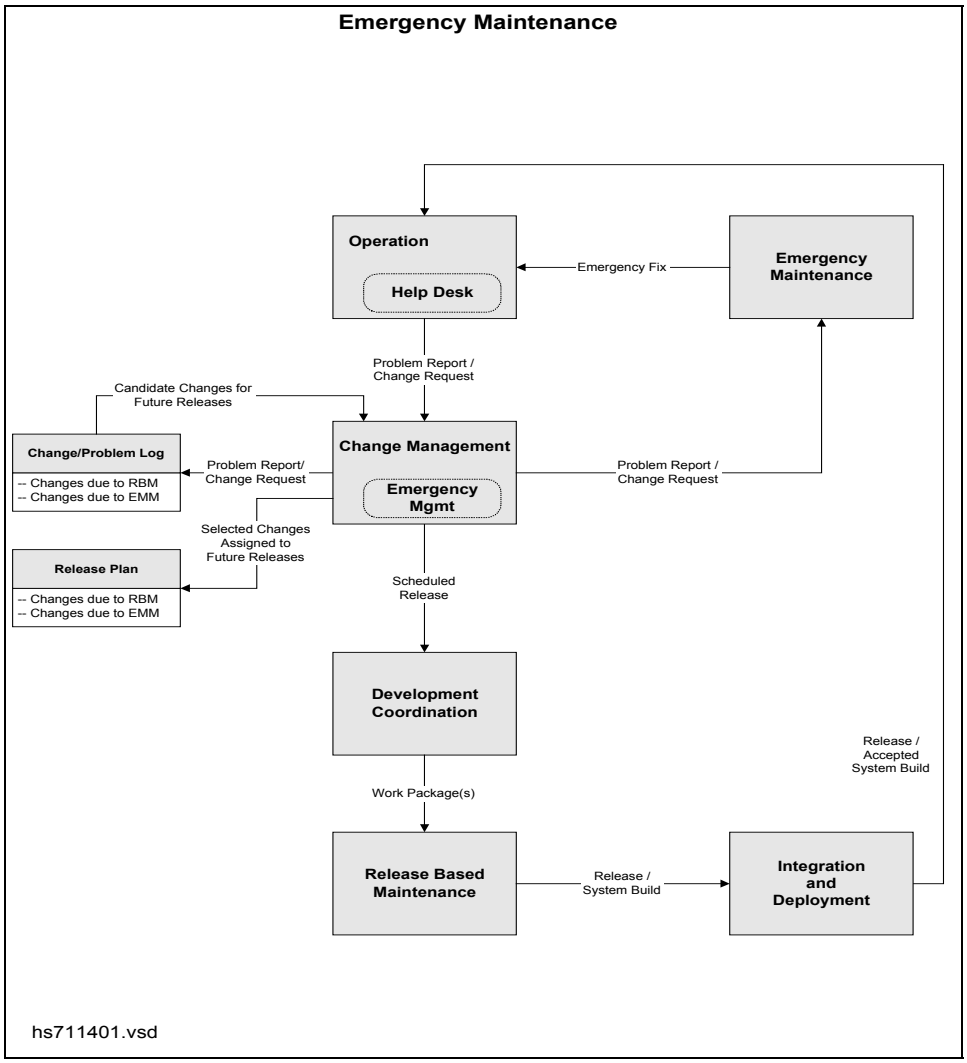


Figure 11–16. Emergency Maintenance in Context

Triggers and results are shown in the process flow diagram (see Figure 11–17). The data flow diagram (see Figure 11–18) illustrates the flow of data among the nine lower level activities:

1. **Understand the Problem**. Study and understand the problem as it has been described by the Operations staff in the Problem Report. Check and complete the information as needed to be able to start a solution approach.
2. **Assess and Classify Problem**. Once the problem is understood, assess the gravity of the problem, the risks, and the impact of performing or omitting an intervention to the system. Classify the problem and decide if an Emergency Maintenance Intervention is required. Determine if the problem can be resolved at the system administration level; if not, proceed with Emergency Maintenance.
3. **Identify Solution Approaches**. With the problem now classified as an Emergency Maintenance case, continue to study the problem to identify alternative solution approaches. Describe the approaches in an Intervention Plan and propose a sequence for choosing the next approach if the prior one has failed.

Intervention Plan. An Intervention Plan documents the emergency fix approaches together with their priority and possible interdependencies. It describes the sequence of different approaches that have been created to solve the emergency problem.

The Intervention Plan also serves as a log that captures the complete history of the emergency intervention and the information that is derived from the approaches that failed. This log provides essential lessons learned for subsequent interventions.

Keep the Intervention Plan short and easy to understand. Remember that it supports the emergency intervention and captures the information about the process.

Attach the Intervention Plan to the Problem Report so that all records of the emergency maintenance are kept together.

4. **Validate Emergency Maintenance Environment**. After identifying the technical, personnel, and organizational resources required for performing the Emergency Maintenance, ensure that all prerequisites are satisfied at all Operations and Emergency Maintenance Team sites.

   At the completion of this activity, the teams and environments at Operation and Emergency Maintenance Team sites are ready for implementing the solution approaches and testing the emergency fix.
5. **Implement and Test Solution**. Start implementing the solution as soon as a solution approach has been selected. To avoid delays, perform short tests of the modified or new code related to the Emergency Fix as early as possible.
6. **Install Emergency Fix**. Make sure that the Operations site is ready for installing the Emergency Fix and that security provisions can be performed. Install the Emergency Maintenance Fix as soon as it has been implemented, all materials are available, and short-tested.
7. **Test Emergency Fix**. Test the Emergency Fix against the problem description under the same conditions as the problem occurred. Execute the fix at the Operations site. Record the findings in the Intervention Plan.
8. **Evaluate Results of Approach**. After the Fix has been tested, evaluate the results of the solution approach for the emergency problem to decide if the Emergency Fix solves the problem sufficiently or if another approach must be taken.
9. **Reassess Emergency Problem and Fix**. After the emergency situation has been resolved, reassess both the problem that caused that emergency and the Emergency Fix. Include both system and business aspects in the assessment. Transfer the Emergency Fix into a Change Proposal and pass it to the Change Management if this is appropriate. The Fix will then be replaced by a release-based change in a later release.

   Emergency fixes are done under severe time restrictions, with the focus on getting the system to operation as quick as possible and not on the quality aspects that apply to regular system evolution as done by Release-Based Maintenance. Furthermore, emergency fixes might be done as patches to run-time modules and not traced back to the source code modules after the system is running again. This situation may cause increased maintenance efforts, and increase the risk of losing the fix in a subsequent system or releases.

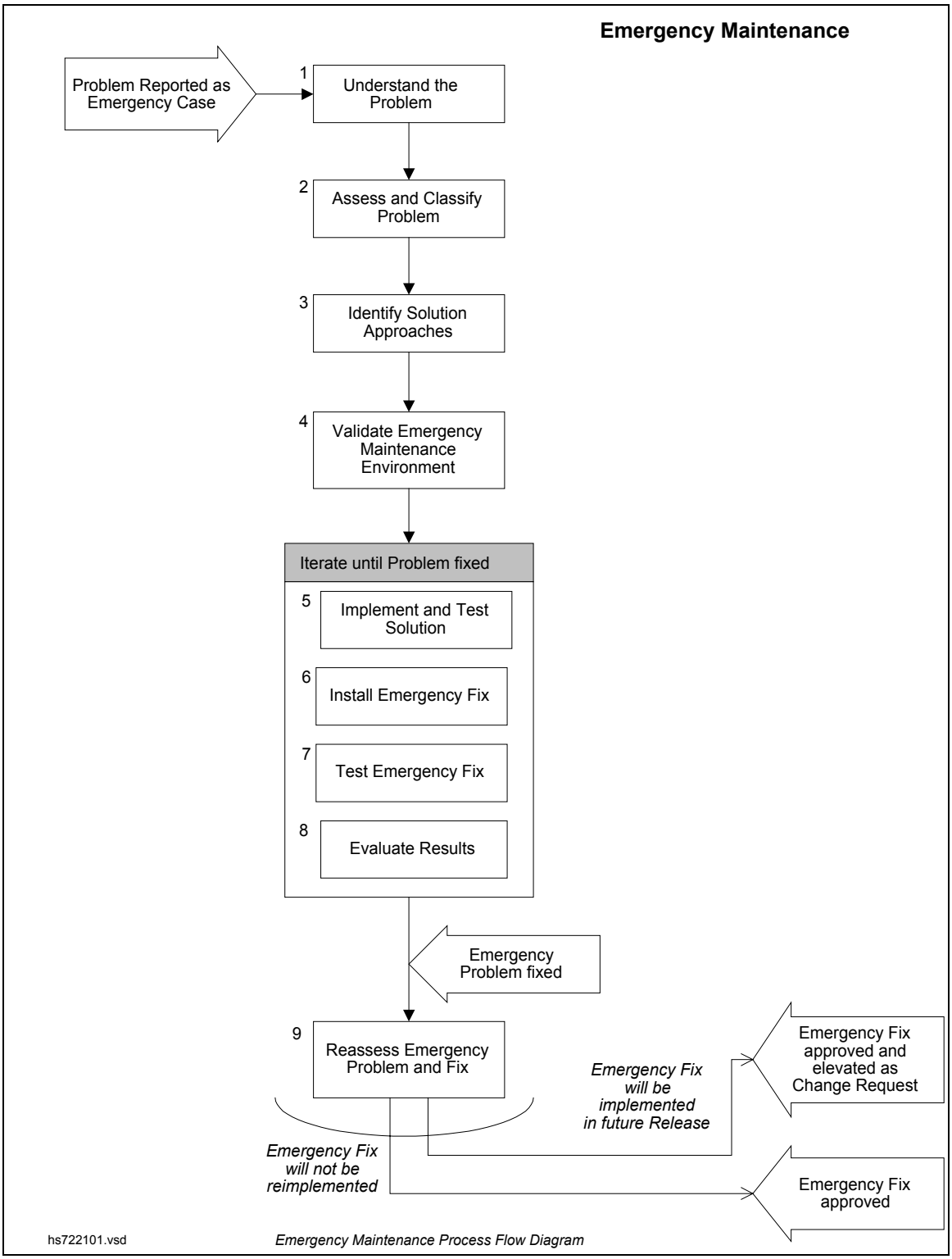


Figure 11–17. Emergency Maintenance, Process Flow Diagram

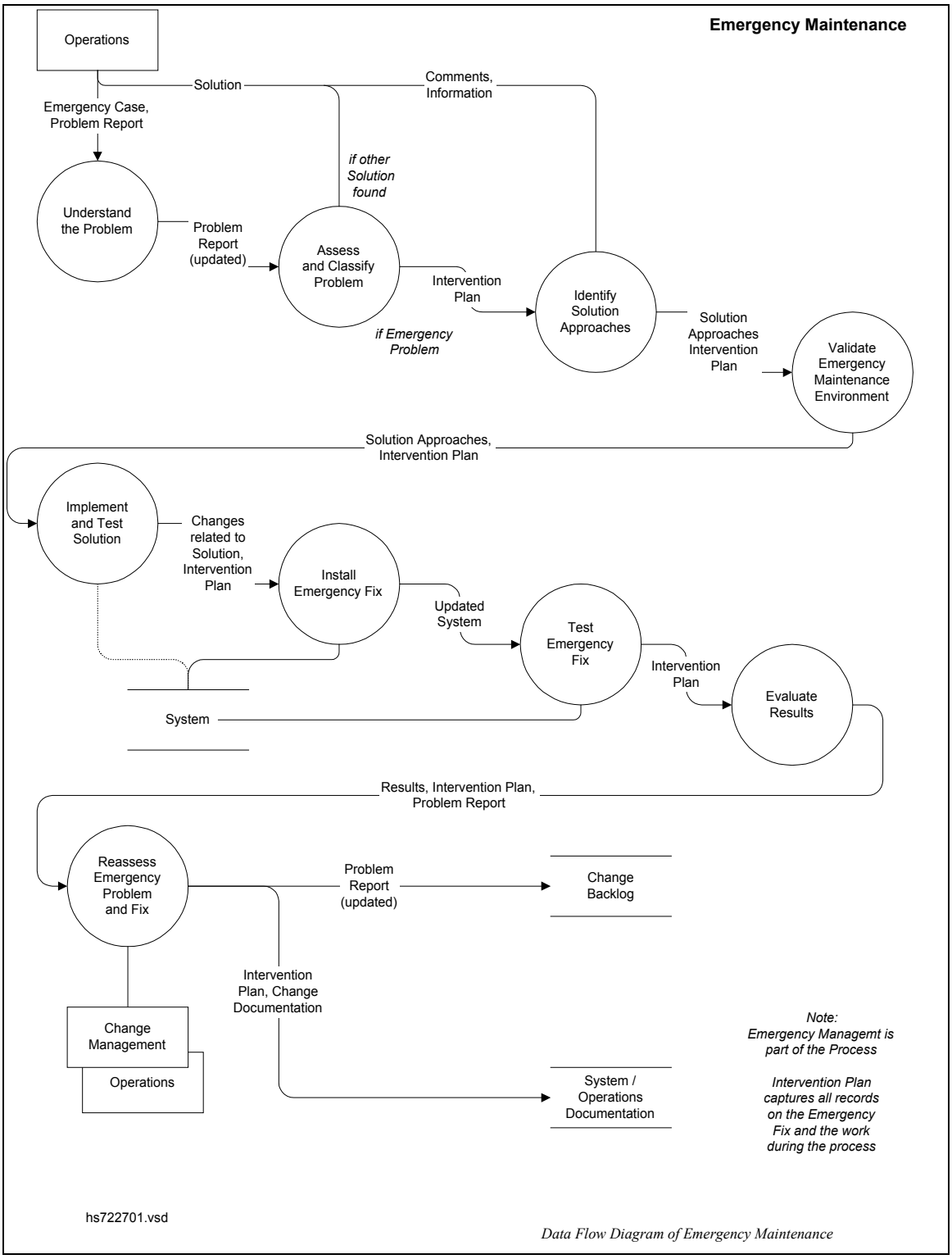


Figure 11–18. Emergency Maintenance, Data Flow Diagram

Table 11–9 associates the nine activities with the roles responsible for their execution.

Table 11–9. Emergency Maintenance, Activity-Role Matrix

Activities	Roles								
Accountable for Execution: Overall Project Manager, Technical Project Manager, Development Team *Approval to Proceed: Business Advocate* *Provides Input: Business SME, Technical SME, Other Representatives or Stakeholders: CM, QA*	OPM	TPM	DT	BA	BSME	TSME	OTHER	CM	QA
1. Understand the problem			P						
2. Assess and classify problem			P						
3. Identify solution approaches			P						
4. Validate emergency maintenance environment			P						
5. Implement and test solution		A	P						
6. Install emergency fix		A	P					R	
7. Test emergency fix			P						
8. Evaluate results of approach		R	P						
9. Reassess emergency problem and fix		P	P					R	

Legend: P = Performs, R = Reviews, A = Approves, S = Supports

Figure 11–15, at the end of the previous section on routine (release-based) maintenance, depicts the flow of a software module through the various software libraries during emergency maintenance.

12. Component 7. Decommission the Solution

12.1 Purpose

The purpose of this component is to deactivate an operational system, which includes:

- Coordinate with the Records Management Branch
- Determine the impact on any other systems
- Remove the system from the operational environment
- Update the inventory

12.2 Roles and Responsibilities

The roles required to perform the activities in the Decommission the Solution component are shown in Table 12–1.

Table 12–1. Component 7 Roles and Responsibilities

Roles	Responsibilities
Overall Project Manager Technical Project Manager Development Team	Accountable for Execution
Business Advocate Executive Sponsor	Approval to Proceed
Business Project Manager Business SME Technical SME Other Representatives or Stakeholders • Quality Assurance Manager • Configuration Management Manager • Records Management	Provides Input

12.3 Entry Criteria

Any of the following events may trigger the initiation of Component 7:

- Service date
- Technology Development Plan project selected
- Installation of a replacement technology item
- Records retention schedule expiration

12.4 Input, Activities, and Outputs

The inputs to this component are as follows:

- Project Action plan
- Software Engineering Notebook
- Installed solution
- Operational Environment
- System Inventory
- Technical Reference Model
- GILS

Figure 12–1 illustrates the flow of the following activities:

1. Review and update plans and announce decommissioning
2. Notify Records Management Branch to review records management requirements
3. Analyze system interfaces and document effect on any other systems
4. Obtain approval to remove the solution from the operational environment
5. Update documentation and records
6. Wait for confirmation and remove the system
7. Obtain final sign-off that decommissioning is complete

As the figure suggests, many of the activities can be performed in parallel. The activities are described in more detail in Section 12.7.

Figure 12–1 also summarizes the outputs of all Component 5 activities in the central data store.

Appendix C includes a summary description of all products of NRC projects, indicates within which components each product is created and updated, and specifies which products are required.

The products, whose standard and form numbers are shown in the figure, are described in more detail in the companion volume *SDLCM Methodology Procedures, Standards, and Forms*.

12.5 Techniques and Tools

The activities of Component 7 are supported by the following techniques.

- Structured Facilitation
- Structured Walkthrough
- Peer Review

Refer to the current *SDLCM Methodology Tool Inventory* for the recommended set of tools.

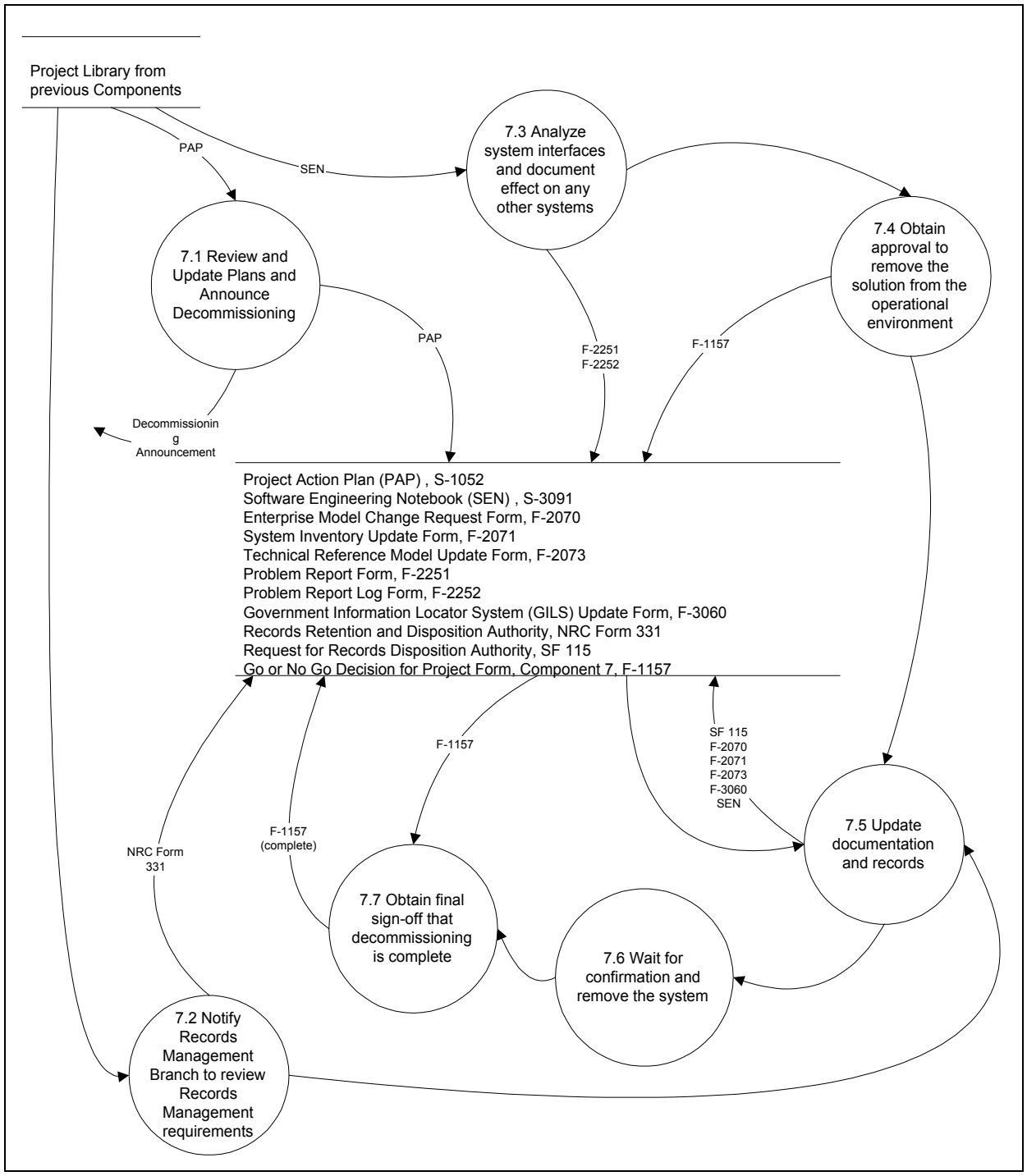


Figure 12–1. Component 7

12.6 Exit Criteria

Component 7 is complete when:

- All critical products have been reviewed (structured walkthrough or peer review)
- Key products have been inspected by QA to conform to project standards
- Products have been placed under CM control
- Information has been shared
- Issues have been resolved
- There is management approval that the decommissioning is complete

12.7 Component 7 Activity Details

The following pages provide detailed activities of Component 7, Decommission the Solution.

Activity 7.1: Review and Update Plans and Announce Decommissioning

See Figure 12–2.

- Review the Project Action Plan, and announce the decommissioning to the entire NRC community. It is important to prepare people to expect decommissioning activities. The system may have users of whom the owner is unaware.
- From the users' point of view, determine what will be done with the software, data, and documentation of the decommissioned system (for example, leave it in the inventory for possible future use, turn it over to a different organization, save it for a specified time period, destroy it). (Analysis of the legal and regulatory Records Management requirements may necessitate alternative disposition of the system.)
- Update the Project Action Plan.

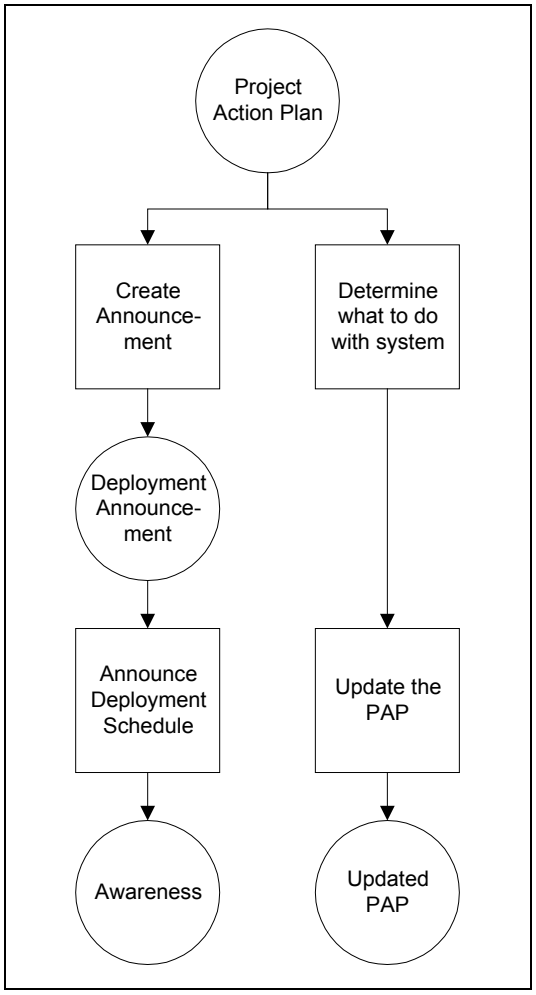


Figure 12–2. Review and Update Plans and Announce Decommissioning

Activity 7.2: Notify Records Management Branch to Review Records Management Requirements

See Figure 12–3.

1. Review copies of the previously submitted Records Management forms in the project library:
   - Information System Description, NA Form 14028
   - Records Retention and Disposition Authority, NRC Form 331
   - Notification of Electronic Information System Design or Modification, NRC Form 616
   - Request for Records Disposition Authority, SF 115
2. Complete a new NRC Form 331 and deliver it to the Records Management as notification that that an application system is about to be decommissioned.
3. After the form has been processed, combine the legal and regulatory requirements with the business and technical requirements (documented in Activity 7.3) and update the documentation in Activity 7.4.

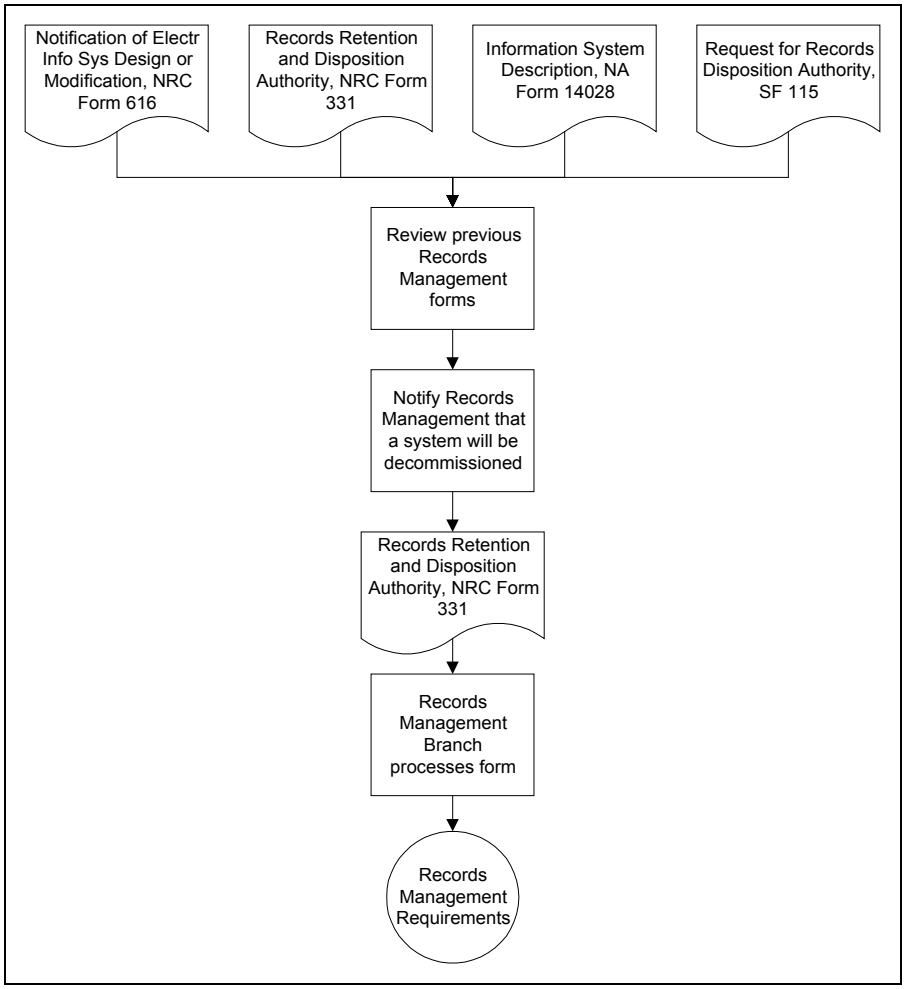


Figure 12–3. Notify Records Management Branch to Review Records Management Requirements

Activity 7.3: Analyze System Interfaces and Document Effect on Any Other Systems

See Figure 12–4. Using information in the Software Engineering Notebook,

1. Analyze the system to determine any external interfaces or dependencies.
2. If other application systems make calls to functionality provided by this system or make use of any data generated by this system, submit Problem Report(s) using Form F–2251 to the CCBs of the affected systems to document the fact that the dependencies must be removed. (NOTE: It is the responsibility of the owners of the affected systems to make the necessary changes or to request that this decommissioning activity be terminated. The decommissioning team must not initiate Activity 7.6 before receiving confirmation that all dependencies have been removed.)

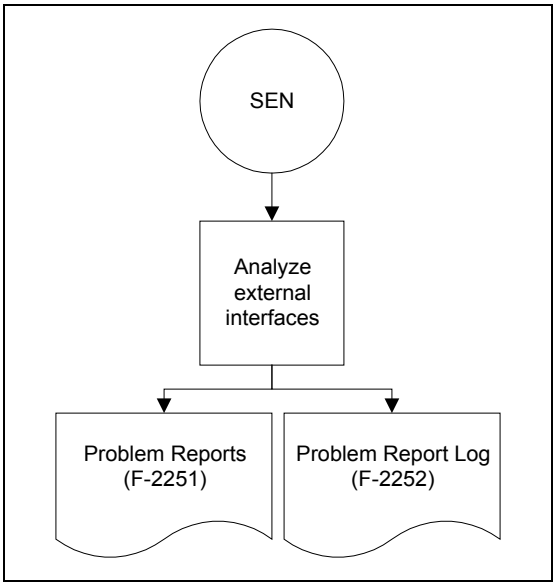


Figure 12–4. Analyze System Interfaces and Document Effect on Any Other Systems

Activity 7.4: Obtain Approval to Remove the Solution from the Operational Environment

See Figure 12–5. Complete the Go or No Go Decision for Project Form, Component 7, and submit it to the Business Advocate to get the user community's approval to remove the solution from the operational environment.

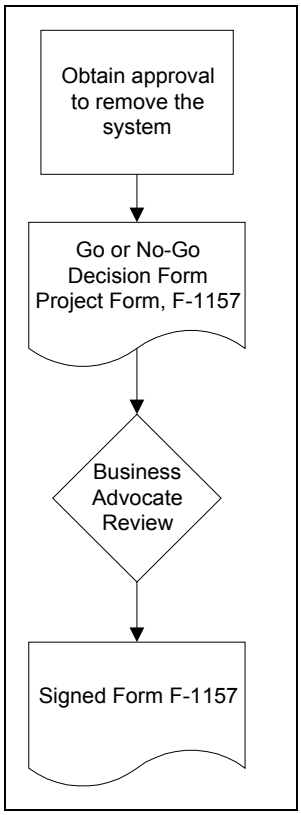


Figure 12–5. Obtain Approval to Remove the Solution from the Operational Environment

Activity 7.5: Update Documentation and Records

See Figure 12–6. After the Records Management requirements are known and the interfaces have been analyzed,

1. Update the system documentation (SEN) unless everything will be destroyed immediately.
2. Update the Enterprise Model, System Inventory, TRM, and GILS.
3. Notify Records Management.
4. Update any affected operational policies and procedures to remove references to the application system about to be removed.

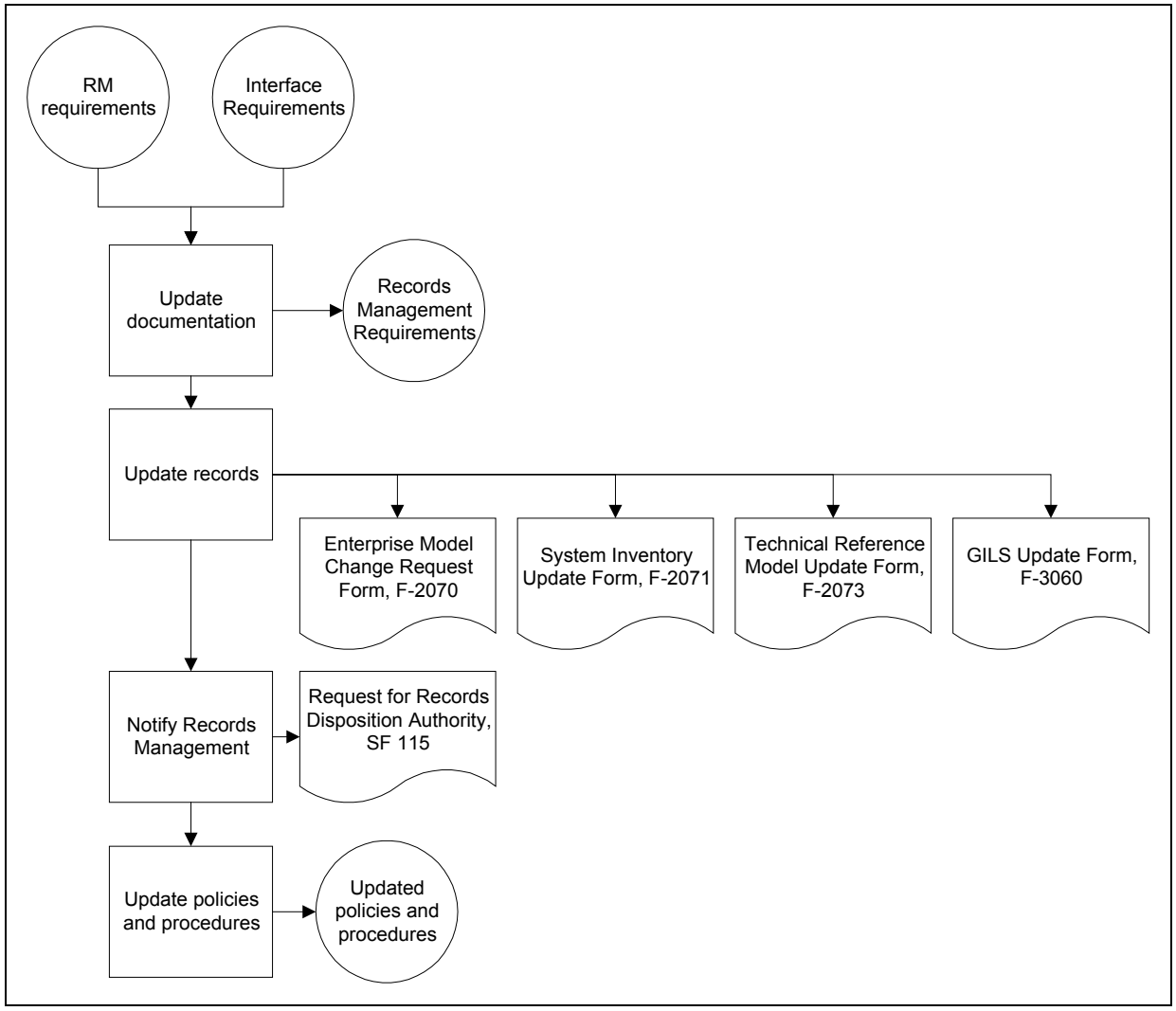


Figure 12–6. Update Documentation and Records

Activity 7.6: Wait for Confirmation and Remove the System

See Figure 12–7.

1. Wait for confirmation that any affected systems have been re-deployed.
2. Using the plans for decommissioning documented in the PAP, remove the system from the operational environment. Archive or destroy related processes and data as documented in the PAP.

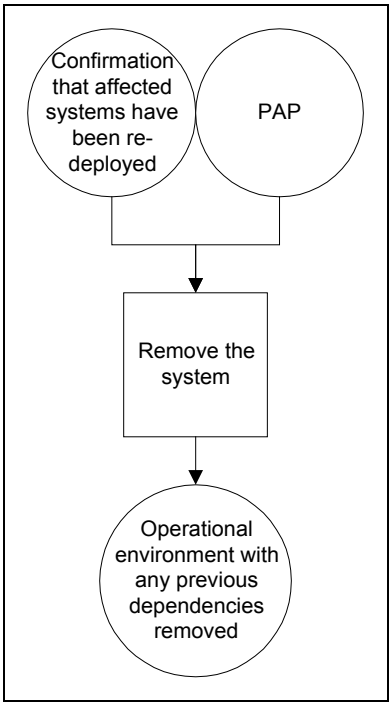


Figure 12–7. Wait for Confirmation and Remove the System

Activity 7.7: Obtain Final Sign-Off That Decommissioning Is Complete

See Figure 12–8. After the system has been removed from the environment and all required documentation has been completed, obtain the final sign-off from the Executive Sponsor that the decommissioning is complete.

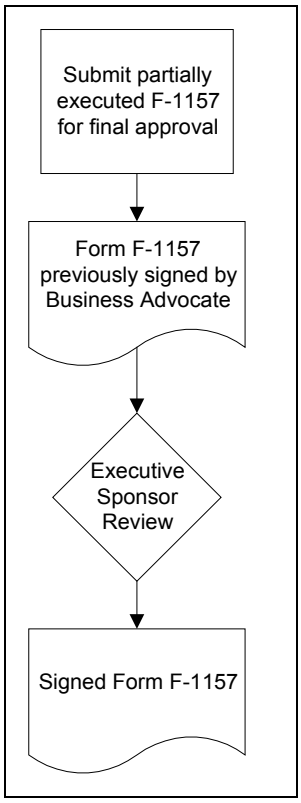


Figure 12–8. Obtain Final Sign-Off That Decommissioning Is Complete

Appendix A. Maintaining the SDLCM Methodology

The SDLCM Methodology is the approach to doing business at the NRC. That approach is described by a set of documents, including this handbook and the companion volume of procedures, standards, and forms.

As discussed in Section 2.4, every project provides an opportunity to improve the process defined by the SDLCM Methodology.

SDLCM Methodology Procedure P–9001 (SDLCM Methodology Change) defines the mechanism for reporting deficiencies—or suggesting improvements—in the methodology or in the documents describing the methodology. To request a change to the SDLCM Methodology or to any part of its documentation set, follow the steps specified in Procedure P–9001.

SDLCM Methodology Form F–9001 (SDLCM Methodology Change Request Form) is the vehicle for documenting any change.

Appendix B. Roles and Responsibilities

This appendix provides a detailed summary of the responsibilities of the various project roles identified in the SDLCM Methodology.

Roles	Responsibilities
Executive Sponsor	• Being the champion of the business change program and the initial source for determining its scope and objectives • Being the chief decision-maker responsible for the business program and being ultimately responsible for the decision to have the process automated or the system upgraded • Being the senior business official (typically an SES) who approves the project and authorizes resources • Having approval authority for budget • Being involved in scoping and high-level decision-making for the project • Empowering Business Advocates to participate fully in the project • Approving requirements definition, authorizing resources, approving design, authorizing deployment, and decommissioning • Providing funding and ensuring long-term support for implementation • Using political influence and corporate knowledge to ensure project success • Approving (go or no-go decisions) at milestones identified in action plans • Monitoring project progress • Resolving sensitive or critical issues as needed • Making policy decisions • Communicating project activities and successes throughout the organization • Communicating to the team external events and situations that may affect the project on a timely basis • Clearing calendars of project team members as needed

Roles	Responsibilities
Overall Project Manager	• Planning the project, allocating and directing the staff and other resources to accomplish project tasks, and maintaining control over the project • Being responsible for project coordination and day-to-day contacts • Having high level knowledge of business and technology • Coordinating and developing the business *and* technical aspects of the project • Being responsible for the deliverables, cost, schedule, and quality of the project • Providing project team leadership to business and technical staff • Coordinating business representation and IS activities (through technical project plan) • Removing obstacles to project success • Coaching, developing, and supporting project team members • Ensuring all team members have appropriate training and experience • Maintaining totally integrated control of the project • Managing all the business aspects of the project • Providing all briefing and status reporting information to all levels of interested staff and management. • Perhaps also playing the roles of the Business Project Manager or the Technical Project Manager, as determined on a project-by-project basis
Business Project Manager	• Sharing responsibility for project coordination and day-to-day contacts with the Overall Project Manager • Having high level knowledge of process and requirements • Representing the sponsoring office (typically non-SES) • Directing and overseeing day-to-day project activities affecting the business view of the project • Coordinating and developing the business aspects of the project • Reporting progress and involving appropriate customer and business experts • Recommending go or no-go decision at each milestone for Executive Sponsor approval • Managing business office resources in the design, engineering, and deployment of the solution • Working with the Technical Project Manager • Selling policy and practice changes • Perhaps also playing the role of the Overall Project Manager, as determined on a project-by-project basis

Roles	Responsibilities
Technical Project Manager	• Being responsible for technical project coordination • Having high-level knowledge of technology, tools, and methodology • Being the technical representative (typically non-SES) • Having responsibility for all IT-related aspects of the project • Managing all technical aspects of the project • Marshaling the Development Team • Working with the Business Project Manager • Managing all technical development of the system • Managing staff and contractors involved in project • Controlling the development schedule • Managing developer resources and the use of the methodology • Coordinating development activities to conform to business requirements • Ensuring that the SDLCM Methodology is followed • Perhaps also playing the role of the Overall Project Manager, as determined on a project-by-project basis

Roles	Responsibilities
Business Advocate	• Being the primary beneficiary or stakeholder • Being the business leader whose business area is affected most by the project and who has the lead role in the direction and use of the application outcome • Driving system requirements • Holding a key role in the business area where the system is being implemented and perhaps being: ⇒ The customer who has a vested interest in the project (that is, a real user) ⇒ The functional area manager responsible for the business area where the solution is required ⇒ The principal user responsible for success of the project • Keeping the vision for the business needs of the project • Being a visionary who identifies problems and recognizes the need for improvement in a business area • Seeking authorization and approval from the Executive Sponsor • Being an enthusiast, a sales person, a resource identifier, and a business verifier • Overcoming project problems in all areas (resources, scoping, design, selling, installation) • Selling the solution upward to executive management and executive sponsorship • Selling the business need downward and outward to all involved with the project (equivalent of private industry program or matrix management) • Ensuring that all business requirements have been addressed by the solution • Approving all project plans and deliverables • Identifying and resolving critical issues • Obtaining approval for new business policies and practices to support the new solution • Participating in the implementation of the solution within his or her own business area • Working closely with development team to ensure compliance with business area requirements • Testing and reporting discrepancies

Roles	Responsibilities
Development Team Individual team roles may include: • Data Modeler • Database Administrator • Knowledge Coordinator • Analyst • JAD Facilitator • Prototype Developer • Designer • Coder • Tester • Configuration Manager • QA representative	• Doing the detailed work on one or more aspects of the system (definition, design, engineering, deployment, service or decommissioning) • Building applications • Developing prototypes • Gathering technical alternatives • Refining, designing, developing, and deploying the system • Gathering, validating, and testing against user and technical requirements • Testing (and resolving issues) • Documenting the requirements, design, software, and operational and maintenance processes • Developing and obtaining approval for all project products • Developing user guides • Controlling project baselines and changes to those baselines • Ensuring that quality is built into the system • Following SDLCM Methodology guidance
Business Subject Matter Expert (SME)	• Advising the Business Advocates concerning business area requirements • Providing detailed functional expertise input and guidance throughout project life cycle • Clarifying business requirements • Providing input to prototype refinements • Providing business area information not available from Business Advocates • Participating in the development by providing the business perspective • Providing input to validate feasibility of prototype or design • Perhaps being an industry expert, a business function expert, or a business process expert
Technical Subject Matter Expert (SME)	• Being an authority who has specialized knowledge of the technical side of the project • Providing technical expertise to the Development Team • Being infrastructure experts, component hardware or software experts, application package experts, or product and technology experts • Working closely with the Development Team to ensure that the technical solution conforms to business requirements • Providing detailed technical expertise • Identifying potential fixes • Providing input that defines or clarifies technical requirements • Advising the Development Team on alternative designs, development approaches, applicable new technologies, etc.
Other Representatives or Stakeholders	• Representing the business needs of non-lead office(s) significantly affected by the system being developed or affected by the outcome of the solution • Providing input and perspective to project plans and deliverables

Appendix C. Products of Projects

This appendix identifies and summarizes all products of projects as specified by NRC's SDLCM Methodology and specifies which products are required. Use this appendix together with Appendix D, which summarizes the activities and identifies the products of each activity, as a cross-reference to the component descriptions in the main body of this handbook.

Section C.1 summarizes the products of NRC projects, organized by the component of the methodology within which the product is initially created.

Section C.2 addresses legacy systems.

C.1 Products of Projects by SDLCM Methodology Component

The tables in this section summarize the products according to the components in which they are initially created. For a convenient listing of all products ordered by product identification numbers, see the Table of Contents of the companion volume of *SDLCM Methodology Procedures, Standards, and Forms*. The standards and forms in that volume represent the products of projects.

The SDLCM Methodology distinguishes among activities associated with (1) new development, (2) enhancement, and (3) maintenance of application systems. See Chapter 2, "Methodology Overview," and Chapter 11, "Component 6. Service the Solution," for more information.

- Tables C–1 through C–5 (products of Components 1 through 5) pertain to new development and enhancement projects.
- Table C–6 (products of Component 6) pertains to operational systems that have been deployed and are under maintenance.
- Table C–7 (products of Component 7) pertains to any operational system that is being decommissioned.

When performing *maintenance* (Component 6), update only the system documentation products that are needed to support further system maintenance. For example, if the software or system tests are modified, then update the Test Plan; if the user interface or functionality is modified, then update the User Guide. Do not update any system documentation products that are not affected by the maintenance change. See also Section C.2.

Table C–1. Products Created within Component 1

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Project Charter (S–1051)	1		The project charter is a management document used to initiate an NRC project that will be developed using the SDLCM Methodology. The project charter identifies the high-level goals and objectives of the project. It specifies the project's key personnel, including the Executive Sponsor and Overall Project Manager, and commits their time and resources. It provides the background, scope, and a high-level approach for its development. It also identifies any constraints and critical success factors necessary to the management of the project.	Required for new development projects. Not required for enhancement projects.
Project Definition and Analysis Document (S–3051)	1	3,4,6	The PDAD clarifies the scope of the project. It contains the system functional requirements, data requirements, an assessment of the current system (if any), an analysis of alternative approaches for satisfying the requirements, and a system operations concept.	Required
Current System Assessment Document (S–3052)	1		Use the Current System Assessment Document to document the results of the assessment of the current system if the assessment is complex and cannot be summarized in the Project Definition and Analysis Document.	Required if not included as part of the PDAD
Project Action Plan (S–1052)	1	2,3,4,5,6,7	The PAP communicates to management, the customer, and project members the overall plan for performing and managing the project from start to end. It consists of two parts. Part 1, the project management plan, defines the activities to be accomplished to satisfy project requirements, provides a master schedule and staffing plan for the performance of these activities, and describes the management and technical approach to accomplishing the project objectives. This part of the PAP is created as an activity within Component 1 and is updated as the project grows and matures. Part 2, the software development plan, provides the detailed activities and schedules for designing, coding, integrating, and testing new, legacy, and COTS software modules to provide the full functionality of the software for the project. It is developed on projects that include software development or integration after project software requirements have been identified as an activity within Component 3. The software development plan portion of the PAP also grows and is updated as the project progresses and matures.	Required

Table C–1. Products Created within Component 1 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Tactical Integration Plan (S–5051)	1	2,3,4,5,6	The TIP serves as notification of the intended deployment of a system or application and provides the time frame for scheduled operation. It provides NRC management with the information necessary to (1) ensure that the level of planning is sufficient to proceed with the system deployment described, (2) assess the impact on other components of the NRC Enterprise Model, and (3) confirm the adequacy of the schedule and budget to complete the deployment and initial operation.	Required
Support Resource Acquisition Request and Commitment Form (F–1061)	1,6		Use this form, if needed, to identify and request personnel and other resources.	Required only if needed
Environment Change Request Form (F–1601)	1,6		Use this form, if needed, to identify and request new or upgraded technology (including tools) that will change the NRC computing environment.	Required only if needed
Notification of Electronic Information System Design or Modification (NRC Form 616)	1,6		Use this form to notify the Records Management Branch that a system is being designed or modified. RM will assign a representative to act as its interface with the Project Manager.	Required
Records Retention and Disposition Authority (NRC Form 331)	1,7		Use this form to notify the Records Management Branch of the potential need to establish or change the disposition of official records.	Required
Information System Description (NA Form 14028)	1		Use this form to provide a description of the system to the Records Management Branch.	Required
Request for Records Disposition Authority (SF 115)	1,7		Use this form to propose a disposition of records. RM will submit the request to NARA.	Required

Table C–1. Products Created within Component 1 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Data Models (S–3151)	1	3,6	Data models may be included in the PDAD, the Logical Design Document, and the Physical Design Document	Required
Process Models (S–3161)	1	3,6	Process models may be included in the PDAD, the Logical Design Document, and the Physical Design Document	Required
Context Diagrams (S–3162)	1	3,6	A context diagram must be included in the PDAD and may be updated and included in the Logical Design Document	Required
Data Flow Diagrams (S–3163)	1	3,6	Data flow diagrams may be included in the PDAD, the Logical Design Document, and the Physical Design Document	Required
Data Dictionary (S–3351)	1	3,6	The data dictionary is a mechanism for defining a system's data elements. Information about the data dictionary may be included in the PDAD, the Logical Design Document, and the Physical Design Document	Required
SDLCM Methodology Deviation or Waiver Form (F–2010)	1		Use this form to request a deviation or waiver from a requirement of the SDLCM Methodology. (Deviation: fulfill the requirement with modifications. Waiver: eliminate the requirement.)	Required only if needed
Enterprise Model Change Request Form (F–2070)	1,3,7		Use this form to report a change to the NRC Enterprise Model	Required
Go or No GO Decision for Project Form, Component 1 (F–1151)	1		Use this form to request approval to proceed with the activities of Component 2, Acquire Support Resources.	Required

Table C–2. Products Created or Updated within Component 2

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Project Action Plan (S–1052)	1	2,3,4,5,6,7	(See Component 1.)	Required
Tactical Integration Plan (S–5051)	1	2,3,4,5,6	(See Component 1.)	Required
Development and Maintenance Environment Products Installation Plan (S–1055)	2	6	The Development and Maintenance Environment Products Installation Plan provides a detailed description of the activities involved in the installation of all resources required to design, engineer, and maintain the system. It defines responsibilities, schedules, risks, and risk mitigation approaches.	Required only if needed
Statement of Work (S–1053)	2		The SOW is the project manager's description of the resources required from the contractor who will provide them. It serves as the basis of a contractual agreement with the supplier for the acquisition of resources required for the project.	Required only if needed
Support Resource Acquisition Strategy Form (F–1062)	2		Use this form to document the strategy for acquiring the needed resources.	Required only if needed
Developer Training Requirements Form (F–7051)	2		Use this form to document the training requirements for project personnel.	Required only if needed
Fully Staffed and Trained Team Form (F–1181)	2		Use this form to document the roles of the project personnel and to document that required training has been completed.	Required
Go or No GO Decision for Project Form, Component 2 (F–1152)	2		Use this form to request approval to proceed with the activities of Component 3, Design the Solution.	Required

Table C–3. Products Created or Updated within Component 3

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Project Action Plan (S–1052)	1	2,3,4,5,6,7	(See Component 1.)	Required
Project Definition and Analysis Document (S–3051)	1	3,4,6	(See Component 1.)	Required
Logical Design Document (S–3171)	3	4,6	The Logical Design Document presents the system architecture at a level of detail sufficient to begin detailed physical design activities. It translates the system's requirements, contained in the PDAD, into the functions to be performed by the hardware, software, and firmware components of the system. It shows how the various components will work together to meet the operational requirements.	Required
Physical Design Document (S–3172)	3	4,6	The Physical Design Document summarizes the results of translating the logical design objects into physical design objects. The physical design objects include a relational schema, a relational table structure diagram, the beginnings of the data definition language, a Data Dictionary, and the screen prototypes.	Required
Tactical Integration Plan (S–5051)	1	2,3,4,5,6	(See Component 1.)	Required
Products Installation and Integration Plan (S–5052)	3	4,5,6	The Products Installation and Integration Plan provides a detailed description of the activities involved in the installation and integration of equipment resources (Commercial or Government off-the-shelf (COTS or GOTS) hardware and software and custom-developed software) required for the deployment of the solution system. It defines responsibilities, schedules, risks, and risk mitigation approaches.	Required if not included in update of TIP
Solution Integration Plan (S–5053)	3	4,5,6	The Solution Integration Plan provides a detailed description of the activities involved in the integrating the hardware and software configuration items (CIs) of the solution system. It also specifies roles and responsibilities and the integration schedule, and identifies any risks, and risk mitigation approaches applicable to the integration process.	Required if not included in update of TIP
Other Systems Integration Plan (S–5054)	3	4,5,6	The Other Systems Integration Plan provides a detailed description of the activities involved in integrating the solution system with other NRC systems, both legacy and new. It defines responsibilities, schedules, risks, and risk mitigation approaches.	Required if not included in update of TIP

Table C–3. Products Created or Updated within Component 3 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Other Integration Issues Plan (S–5055)	3	4,5,6	The Other Integration Issues Plan provides a detailed description of the activities involved in accomplishing any integration issues not covered by other plans. It defines responsibilities, schedules, risks, and risk mitigation approaches.	Required if not included in update of TIP
Conversion Plan (S–1054)	3	4,5,6	The Conversion Plan provides a detailed description of what data will be converted, where the data will come from, and how the supporting business process will be organized, staffed, and scheduled.	Required if not included in update of TIP
Security Plan (S–1056)	3	4,5,6	A Security Plan describes the administrative and technical means to design security into a system. It defines the security policies and details how they are to be implemented. It identifies the roles and responsibilities, as well as the products, activities, and schedules.	Required if not included in update of TIP
Test Plan (S–5151)	3	4,6	The Test Plan maps specific tests to the specific requirements as originally stated in the PDAD (and then carried forward in the logical and physical designs) and describes the approach for performing each of the specific tests.	Required
Integrated Education, Training, and Reference Materials (S–7052)	3	6	The Integrated Education, Training, and Reference Materials summarize the results of the training design activities.	Required
Software Engineering Notebook (S–3091)	3	4,5,6,7	The SEN is the only major product that is not itself a document. It is actually an implementation workbook that consolidates the information pertinent to a software element (or set of software elements). It also ensures ready access to complete and up-to-date information for modification and auditing purposes. Only the most current information about software development and testing is maintained in each SEN. To reduce duplication, SENs do not contain information provided in other documents or data sets. Instead the SEN provides a pointer reference to the relevant document(s). Whenever possible, the SENs should provide pointer references to on-line libraries and files rather than maintaining hard copy material.	Required
External Systems Interface Diagrams (S–3164)	3		An external systems interface diagram is a high-level data flow diagram that shows an application with its interfaces to existing or new computer systems or agents. It illustrates process, external agents, external interfaces, and shared external databases. The diagram may be included in the LDD and PDD, if needed.	Required in LDD and PDD if there are interfaces to external systems

Table C–3. Products Created or Updated within Component 3 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Data Models (S–3151)	1	3,6	(See Component 1.)	Required
Process Models (S–3161)	1	3,6	(See Component 1.)	Required
Context Diagrams (S–3162)	1	3,6	(See Component 1.)	Required
Data Flow Diagrams (S–3163)	1	3,6	(See Component 1.)	Required
Data Dictionary (S–3351)	1	3,6	(See Component 1.)	Required
Enterprise Model Change Request Form (F–2070)	1,3,7		(See Component 1.)	Required
Technical Reference Model Update Form (F–2073)	3,5,7		Use this form to report a change to the NRC Technical Reference Model	Required
Go or No GO Decision for Project Form, Component 3 (F–1153)	3		Use this form to request approval to proceed with the activities of Component 4, Engineer the Solution.	Required

Table C–4. Products Created or Updated within Component 4

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Project Action Plan (S–1052)	1	2,3,4,5,6,7	(See Component 1.)	Required
Project Definition and Analysis Document (S–3051)	1	3,4,6	(See Component 1.)	Required
Tactical Integration Plan (S–5051)	1	2,3,4,5,6	(See Component 1.)	Required
Installation Instructions (S–5252)	4	5,6	The Installation Instructions document provides a detailed description of the activities involved in the installation of a new or enhanced NRC system and the equipment resources, both hardware and software, required for its support in the production environment.	Required if not included in update of TIP
Logical Design Document (S–3171)	3	4,6	(See Component 3.)	Required
Physical Design Document (S–3172)	3	4,6	(See Component 3.)	Required
Test Plan (S–5151)	3	4,6	(See Component 3.)	Required
Operational Support Guide (S–6151)	4	6	The Operational Support Guide provides guidance to operators and system support personnel on how to support users by performing specific tasks in a timely manner. It includes not only servicing users on an event-driven basis, but also performing some tasks periodically.	Required
User Training and Orientation Plan (S–7053)	4	6	The User Training and Orientation Plan provides a detailed plan for assessing the skills of users, providing training and orientation to enable users to operate the new system, specifying the budget and training schedule, and assessing the effectiveness of the training program.	Required
User Guide (S–6051)	4	6	The User Guide provides guidance on how to use the system effectively and efficiently.	Required
On-Line Help System and Tutorials (S–6052)	4	6	The On-Line Help System and Tutorials document is a brief overview of the on-line help and tutorials that are available.	Required
Software Engineering Notebook (S–3091)	3	4,5,6,7	(See Component 3.)	Required

Table C–4. Products Created or Updated within Component 4 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Network Integration Diagrams (S–5056)	4		A set of network diagrams depicts the network support required by the logical design of the system. The form of the diagrams will be determined by the size and complexity of the system being developed or enhanced. The diagrams are included in the TIP.	Required in the TIP if the system runs on a network
Target Users Capability Upgrade Request Form (F–6054)	4,6		Use this form to request capability upgrades for targeted users.	Required only if needed
Solution Modules Ready for Deployment Form (F–4051)	4,6		Use this form to identify the modules of the solution for configuration management.	Required only if needed
Solution Rollout Support Ready for Deployment Form (F–4052)	4,6		This form is a final checklist to ensure that all preparations have been made and that support is in place to begin deployment of the project or system to the site indicated on the form. The form is keyed to the activities identified in the Tactical Integration Plan, Section 2, Rollout Plan, Subsections 2.1–2.16. It refers to those activities that must be completed or ready to be completed at the host site, *not* to the activities required to prepare and test the software, system, or documentation at the development site.	Required only if needed
Change Log Form (F–2561)	4,5,6		The configuration management office uses this form to maintain a record of all change proposals.	Required only if Change Proposals have been submitted
Change Proposal Form (F–2502)	4,5,6		Use this form to request a change to a baselined system.	Required only if changes are needed
Go or No GO Decision for Project Form, Component 4 (F–1154)	4		Use this form to request approval to proceed with the activities of Component 5, Deploy the Solution.	Required

Table C–5. Products Created or Updated within Component 5

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Project Action Plan (S–1052)	1	2,3,4,5,6,7	(See Component 1.)	Required
Tactical Integration Plan (S–5051)	1	2,3,4,5,6	(See Component 1.)	Required
Change Proposal Form (F–2502)	4,5,6		(See Component 4.)	Required only if changes are needed
Problem Report Form (F–2251)	5,6,7		Use this form to report a problem with a deployed system.	Required only if a problem is identified
Problem Report Log Form (F–2252)	5,6,7		The configuration management office uses this form to maintain a record of all Problem Reports.	Required only if Problem Reports have been submitted
Software Engineering Notebook (S–3091)	3	4,5,6,7	(See Component 3.)	Required
System Inventory Update Form (F–2071)	5,7		Use this form to submit an update to the System Inventory	Required
Technical Reference Model Update Form (F–2073)	3,5,7		(See Component 3.)	Required
Government Information Locator System (GILS) Update Form (F–3060)	5,7		Use this form to submit an update to the GILS.	Required
Installation Report Form (F-5254)	5,6		Use this form to report the results of installing the system.	Required only if needed
Rollout Report Form (F–5255)	5,6		Use this form to report on the readiness for the system to be moved to operational status.	Required only if needed
Performance Report Form (F–5256)	5		Use this form to report that the system is ready for acceptance testing.	Required only if needed
Deployment Recommendations for Fix Form (F–5257)	5		Use this form to recommend changes that should be made prior to acceptance testing.	Required only if needed

Table C–5. Products Created or Updated within Component 5 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Customer Satisfaction Form (F–6055)	5,6		Use this form to document customer satisfaction with the system.	Required
Final User Signoff Form (F–6056)	5,6		Use this form as a delivery checklist and to document user acceptance.	Required
Trained Users Form (F–7151)	5		Use this form to document that the users have been trained to operate the system.	Required only if needed
Go or No GO Decision for Project Form, Component 5 (F–1155)	5		Use this form to request approval to proceed with the activities of Component 6, Service the Solution.	Required

Table C–6. Products Created or Updated within Component 6

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Project Action Plan (S–1052)	1	2,3,4,5,6,7	(See Component 1.)	Required
Project Definition and Analysis Document (S–3051)	1	3,4,6	(See Component 1.)	Required
Tactical Integration Plan (S–5051)	1	2,3,4,5,6	(See Component 1.)	Required
Products Installation and Integration Plan (S–5052)	3	4,5,6	(See Component 3.)	Required if not included in update of TIP
Solution Integration Plan (S–5053)	3	4,5,6	(See Component 3.)	Required if not included in update of TIP
Other Systems Integration Plan (S–5054)	3	4,5,6	(See Component 3.)	Required if not included in update of TIP
Other Integration Issues Plan (S–5055)	3	4,5,6	(See Component 3.)	Required if not included in update of TIP
Conversion Plan (S–1054)	3	4,5,6	(See Component 3.)	Required if not included in update of TIP
Security Plan (S–1056)	3	4,5,6	(See Component 3.)	Required if not included in update of TIP
Installation Instructions (S–5252)	4	5,6	(See Component 4.)	Required if not included in update of TIP

Table C–6. Products Created or Updated within Component 6 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Development and Maintenance Environment Products Installation Plan (S–1055)	2	6	(See Component 2.)	Required
Logical Design Document (S–3171)	3	4,6	(See Component 3.)	Required
Physical Design Document (S–3172)	3	4,6	(See Component 3.)	Required
Test Plan (S–5151)	3	4,6	(See Component 3.)	Required
Integrated Education, Training, and Reference Materials (S–7052)	3	6	(See Component 3.)	Required
Software Engineering Notebook (S–3091)	3	4,5,6,7	(See Component 3.)	Required
Operational Support Guide (S–6151)	4	6	(See Component 4.)	Required
User Training and Orientation Plan (S–7053)	4	6	(See Component 4.)	Required
User Guide (S–6051)	4	6	(See Component 4.)	Required
On-Line Help System and Tutorials (S–6052)	4	6	(See Component 4.)	Required

Table C–6. Products Created or Updated within Component 6 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Data Models (S–3151)	1	3,6	(See Component 1.)	Required
Process Models (S–3161)	1	3,6	(See Component 1.)	Required
Context Diagrams (S–3162)	1	3,6	(See Component 1.)	Required
Data Flow Diagrams (S–3163)	1	3,6	(See Component 1.)	Required
Data Dictionary (S–3351)	1	3,6	(See Component 1.)	Required
Support Resource Acquisition Request and Commitment Form (F–1061)	1,6		(See Component 1.)	Required only if needed
Environment Change Request Form (F–1601)	1,6		(See Component 1.)	Required only if needed
Notification of Electronic Information System Design or Modification (NRC Form 616)	1,6		(See Component 1.)	Required
Solution Modules Ready for Deployment Form (F–4051)	4,6		(See Component 4.)	Required only if needed
Solution Rollout Support Ready for Deployment Form (F–4052)	4,6		(See Component 4.)	Required only if needed
Change Log Form (F–2561)	4,6		(See Component 4.)	Required only if Change Proposals have been submitted
Change Proposal Form (F–2502)	4,5,6		(See Component 4.)	Required only if changes are needed
Installation Report Form (F-5254)	5,6		(See Component 5.)	Required only if needed

Table C–6. Products Created or Updated within Component 6 (continued)

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Problem Report Form (F–2251)	5,6,7		(See Component 5.)	Required only if a problem is identified
Problem Report Log Form (F–2252)	5,6,7		(See Component 5.)	Required only if Problem Reports have been submitted
Installation Report Form (F-5254)	5,6		(See Component 5.)	Required only if needed
Rollout Report Form (F–5255)	5,6		(See Component 5.)	Required only if needed
Target Users Capability Upgrade Request Form (F–6054)	4,6		(See Component 4.)	Required only if needed
Customer Satisfaction Form (F–6055)	5,6		(See Component 5.)	Required
Final User Signoff Form (F–6056)	5,6		(See Component 5.)	Required
Go or No GO Decision for Project Form, Component 6 (F–1156)	6		Use this form to request approval to terminate maintenance support for an application system.	Required

Table C–7. Products Created or Updated within Component 7

Product Name	Created in Component	Updated in Components	Product Summary	Required Product?
Project Action Plan (S–1052)	1	2,3,4,5,6,7	(See Component 1.)	Required
Software Engineering Notebook (S–3091)	3	4,5,6,7	(See Component 3.)	Required
Records Retention and Disposition Authority (NRC Form 331)	1,7		(See Component 1.)	Required
Request for Records Disposition Authority (SF 115)	1,7		(See Component 1.)	Required
Enterprise Model Change Request Form (F–2070)	1,3,7		(See Component 1.)	Required
System Inventory Update Form (F–2071)	5,7		(See Component 5.)	Required
Technical Reference Model Update Form (F–2073)	3,5,7		(See Component 3.)	Required
Government Information Locator System (GILS) Update Form (F–3060)	5,7		(See Component 5.)	Required
Problem Report Form (F–2251)	5,6,7		(See Component 5.)	Required only if a problem is identified
Problem Report Log Form (F–2252)	5,6,7		(See Component 5.)	Required only if Problem Reports have been submitted
Go or No GO Decision for Project Form, Component 7 (F–1157)	7		Use this form to document that the system has been decommissioned.	Required

C.2 Legacy System Documentation

The available documentation set for unmodified legacy systems developed prior to the introduction of NRC's SDLCM Methodology typically includes:

- User Guide
- Systems documentation file (equivalent to the Software Engineering Notebook)

Following are the names of other documents that may also be included with legacy systems built prior to NRC's development of the SDLCM Methodology:

- Scoping Document
- Security System Plans
- Programmers Guide
- Operations Manual
- Process documents

When performing maintenance activities on a legacy system to correct errors in existing functionality, it is not necessary to produce all products required by the methodology for new development projects. Update the existing system documentation products in which changes are needed to support further system maintenance. If documentation is not available for a legacy system, see Appendix E for a summary of the process for transition of legacy systems from NRC to a contractor for life-cycle management.

When a legacy system is enhanced with new functionality, all documentation must be produced to facilitate, and reduce the cost and effort associated with, future maintenance activities.

Appendix D. Activity Summary

D.1 Component Review

This appendix summarizes the activities of the seven components of NRC's SDLCM Methodology and specifies which activities are required. Use this appendix together with Appendix C, which summarizes the products of NRC projects, as a cross-reference to the component descriptions in the main body of this handbook.

This appendix provides only *summary* checklists of the high-level activities described in the main body of this handbook. See Chapter 6 through Chapter 12 for details. Use the checklists to identify the activities that will be performed and the products that will be produced for a specific project.

The SDLCM Methodology distinguishes among activities associated with (1) new development, (2) enhancement, and (3) maintenance of application systems. See Chapter 2, "Methodology Overview," and Chapter 11, "Component 6. Service the Solution," for more information. Figure D–1 shows the component structure introduced in Chapter 2 and relates that structure to the sections of this appendix.

- Section D.2 summarizes the activities required for development of new application systems or enhancement of existing systems (Components 1–5).
- Section D.3 summarizes the activities performed for routine and emergency maintenance (Component 6).
- Section D.4 summarizes the activities required when decommissioning any operational system (Component 7).

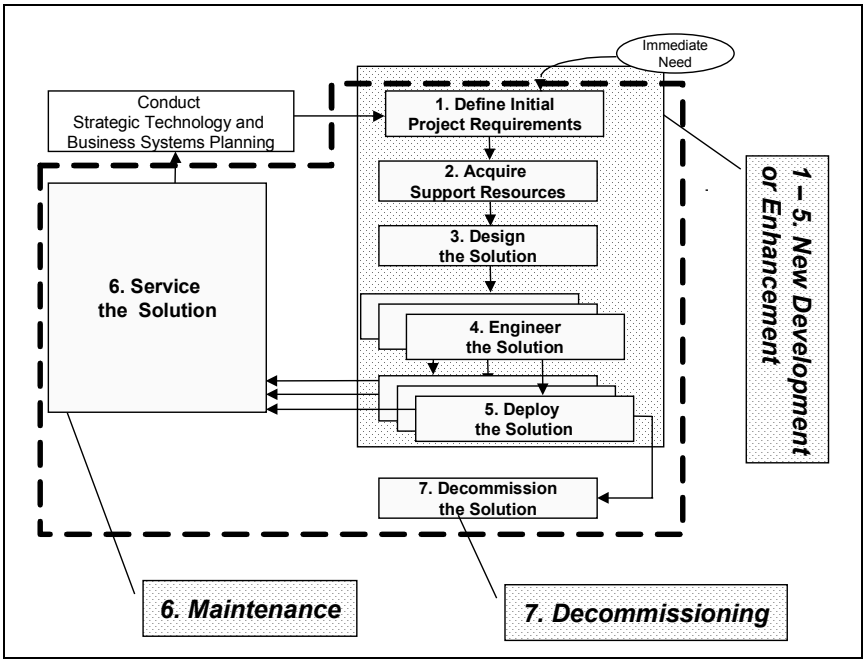


Figure D–1. Review of Component Structure

D.2 New Development and Enhancement

This section summarizes the activities performed for new development and enhancement of application systems. Chapters 6 through 10 provide the details. See Appendix C for a summary of the products that must be produced.

The five tables in this section can be used as checklists for tailoring the SDLCM Methodology to the specific needs of new development and enhancement projects.

Remember: activities are not sequential steps. Although some activities cannot be started until a predecessor activity is complete, others can be performed concurrently. The diagrams in the main body of this handbook clarify the dependencies among activities.

Table D–1. Checklist for Component 1

	1. Define Initial Project Requirements (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	1.1	Clearly identify an information management problem.	Project Charter	S–1051	X	
☐	1.2	Clarify the technical scope of the problem.	Project Definition and Analysis Document • Section 1.3 Scope	S–3051	X	X
☐	1.3	Notify Records Management Branch to Review Requirements.	Government Forms	NRC Form 616 NRC Form 331 NA Form 14028 SF 115	X	
☐	1.4	Identify initial functional and data requirements.	Project Definition and Analysis Document • Section 1.3 Scope • Section 3 System Requirements Specification (SRS) • Section 3.3 Operational Requirements • Section 3.5.2 Records Management Requirements • Section 4 Data Requirements • Section 4.1 Entity List and Definitions • Section 4.2 Context Diagram • Section 7.2.3 User Environment Supporting Standards Data Models Process Models Context Diagrams Data Flow Diagrams Data Dictionary Enterprise Model Change Request Form	S–3051 S–3151, S–3161, S–3162, S–3163, S–3351 F–2070	X	X

	1. Define Initial Project Requirements (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	1.5	Analyze alternative solutions.	Project Definition and Analysis • Section 5 Current System Assessment (*or* Current System Assessment Document) Project Definition and Analysis • Section 7 Systems Operations Concept	S–3051 (S–3052)	X	X
☐	1.6	Plan how to manage the project.	Project Action Plan • Section 2 Part 1—Project Management Plan	S–1052	X	X
			SDLCM Methodology Deviation or Waiver Form	F–2010		
☐	1.7	Review the toolkit	Project Action Plan • Section 3.3.1 Activities, Tools, and Products	S–1052	X	X
☐	1.8	Develop a support resource request.	Project Action Plan Section 2.2.3 Project Organization Support Resources Acquisition Request Form	S–1052 F–1061	X	X
			Environment Change Request Form	F–1601		
☐	1.9	Plan for deployment.	Tactical Integration Plan • Sections for which information can be specified.	S–5051	X	X
☐	1.10	Review the project to make a Go or No-Go decision.	Go or No-Go Decision Form	F–1151	X	X

Table D–2. Checklist for Component 2

	2. Acquire Support Resources					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	2.1	Specify the work to be done	Statement of Work	S–1053		
☐	2.2	Staff the project	Project Action Plan • Section 2.2.3 Project Organization	S–1052	X	X
☐	2.3	Train the staff	Developer Training Requirements Form	F–7051		
			Fully Staffed and Trained Team Form	F–1181	X	X
☐	2.4	Acquire and install other required resources	Development and Maintenance Environment Products Installation Plan Support Resource Acquisition Strategy Form	S–1055 F–1062		
☐	2.5	Update the Project Action Plan	Project Action Plan • Section 2.2 Project Performance Plan • Section 2.3 Risk Management • Part 1 other sections that require changes	S–1052	X	X
☐	2.6	Continue Deployment Planning	Tactical Integration Plan • Provide additional information and update any that has changed.	S–5051	X	X
☐	2.7	Review the project to make a Go or No-Go decision.	Go or No-Go Decision Form	F–1152	X	X

Table D–3. Checklist for Component 3

	3. Design the Solution (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	3.1	Analyze requirements and perform high level design	Project Definition and Analysis Document • Updates to any information that has changed.	S–3051	X	X
			Logical Design Document	S–3171		
			Physical Design Document	S–3172		
			Supporting Standards Data Models Process Models Context Diagrams Data Flow Diagrams Data Dictionary	 S–3151, S–3161, S–3162, S–3163, S–3351		
			Software Engineering Notebook	S–3091	X	X
			Enterprise Model Change Request Form	F–2070	X	X
☐	3.2	Plan solution integration	Solution Integration Plan	S–5053	X	X
			or Tactical Integration Plan • Section 1.3	S–5051		
☐	3.3	Design training materials	Integrated Education, Training, and Reference Materials	S–7052	X	
☐	3.4	Establish test approach	Test Plan	S–5151	X	X
☐	3.5	Update the Project Action Plan	Project Action Plan • Section 3 Part 2 — Software Development Plan • (Updates to any information that has changed.)	S–1052	X	X

	3. Design the Solution (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	3.6	Continue deployment planning	Tactical Integration Plan • Provide any additional deployment information and update any that has changed. *or provide:*	S–5051	X	X
			Conversion Plan	S–1054		
			Security Plan	S–1056		
			Products Installation and Integration Plan	S–5052		
			Other Systems Integration Plan	S–5054		
			Other Integration Issues Plan	S–5055		
			Technical Reference Model Change Request Form	F–2073	X	X
☐	3.7	Review the project to make a Go or No-Go decision.	Go or No-Go Decision Form	F–1153	X	X

Table D–4. Checklist for Component 4

	4. Engineer the Solution (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	4.1	Maintain the change log	Change Log Form Change Proposal Form	F–2561 F–2502	X	
☐	4.2	Engineer the detailed design	Project Definition and Analysis Document Logical Design Document Physical Design Document	S–3051 S–3171 S–3172	X	X
☐	4.3	Build the solution	Software Engineering Notebook	S–3091	X	X
☐	4.4	Integrate the solution	Solution Modules Ready for Deployment Form (Integrated Solution)	F–4051	X	X
☐	4.5	Test the solution	Test Plan • Section 4 Testing (Actual Results)	S–5151	X	X
☐	4.6	Create the rollout strategy and continue deployment planning	Tactical Integration Plan • Section 2 Rollout Plan • Installation Instructions (or as an appendix to the TIP) • Network Integration Diagrams (in TIP) • Provide additional TIP information and update any that has changed. Target Users Capability Upgrade Request Form Solution Rollout Support Ready for Deployment Form	S–5051 S–5252 S–5056 F–6054 F–4052	X	X
☐	4.7	Build the user material	User Guide Online Help System and Tutorials Operational Support Guide User Training and Orientation Plan	S–6051 S–6052 S–6151 S–7053	X	X

	4. Engineer the Solution (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	4.8	Update the Project Action Plan	Project Action Plan • Updates to any information that has changed.	S–1052	X	X
☐	4.9	Review the project to make a Go or No-Go decision.	Go or No-Go Decision Form	F–1154	X	X

Table D–5. Checklist for Component 5

	5. Deploy the Solution (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	5.1	Review and update plans and announce deployment	Project Action Plan Tactical Integration Plan • Updates to any information that has changed. Deployment Announcement	S–1052 S–5051	X	X
☐	5.2	Validate and upgrade the environment	(Target environment)		X	X
☐	5.3	Install the solution	Installation Report Form	F–5254	X	X
☐	5.4	Test the installed solution	(Tested solution)		X	X
☐	5.5	Analyze the test results	Problem Report Form Problem Report Log Form Performance Report Form Deployment Recommendations for Fix Form	F–2251 F–2252 F–5256 F–5257	X	X
☐	5.6	Train the users	Change Proposal Form Problem Report Form Trained Users Form	F–2502 F–2251 F–7151		
☐	5.7	Acceptance test the solution	Software Engineering Notebook Customer Satisfaction Form Final User Signoff Form	S–3091 F–6055 F–6056	X	X
			Change Proposal Form Problem Report Form	F–2502 F–2251		
☐	5.8	Cut over to production environment	Rollout Report Form (Production System)	F–5255	X	X

	5. Deploy the Solution (2 pages)					
	Activities		Products		Activity Req'd	
☑	Act. ID	Activity	Products	Standards or Forms	New Develop-ment	Enhance-ment
☐	5.9	Update policies and procedures	System Inventory Update Form Technical Reference Model Update Form GILS Update Form (Updated system-specific policies and procedures)	F–2071 F–2073 F–3060	X	
☐	5.10	Review the project to make a Go or No-Go decision.	Go or No-Go Decision Form	F–1155	X	X

D.3 Maintenance

This section summarizes the activities performed for routine and emergency maintenance. Chapter 11, "Component 6. Service the Solution," provides the details.

Either a Change Proposal or a Problem Report initiates system maintenance. The Change Management activities are always required. Most changes will be handled as routine. Only true emergencies should be treated as described in Section 11.5.

The products are not specified, because they are strongly dependent on the nature of the maintenance change. See Appendix C for a product summary.

Table D–6. Checklist for Component 6

	6. Service the Solution				
	Activities		Products		Activity Req'd
☑	☑	Activity	Products	Standards or Forms	Maintenance
☑	**Change Management**				X
	☐	Establish change management			X
	☐	Control system changes			X
	☐	Track and report system changes			X
Select either routine or emergency maintenance.					
☐	**Routine Maintenance**				
	☐	**Release Management**			
	☐	Plan the release			
	☐	Coordinate release changes			
	☐	Track release changes			
	☐	Track releases			
	☐	**Release-Based Maintenance**			
☐	**Emergency Maintenance**				
Select the following if the system will no longer be supported.					
☐	**Terminate Maintenance**				
	☐	Review the project to make a Go or No-Go decision.	Go or No-Go Decision Form	F–1156	X

D.4 Decommissioning

This section summarizes the activities required when decommissioning any operational system. See Chapter 12, "Component 7. Decommission the Solution," for details.

Table D–7. Checklist for Component 7

	7. Decommission the Solution				
	Activities		Products		Activity Req'd
☑	Act. ID	Activity	Products	Standards or Forms	All Systems
☐	7.1	Review and update plans and announce decommissioning	Project Action Plan (Updates to any information that has changed.) Decommissioning Announcement	S–1052	X
☐	7.2	Notify Records Management Branch to review Records Management requirements	Records Retention and Disposition Authority	NRC Form 331	X
☐	7.3	Analyze system interfaces and document effect on any other systems (It is the responsibility of the owners of other systems to respond to the PRs if necessary.)	Problem Report Form Problem Report Log Form	F–2251 F–2252	X
☐	7.4	Obtain approval to remove the solution from the operational environment	Go or No-Go Decision Form (Business Advocate signature)	F–1157	X
☐	7.5	Update documentation and records	Request for Records Disposition Authority Enterprise Model Change Request Form System Inventory Update Form Technical Reference Model Update Form GILS Update Form Software Engineering Notebook (Updated system-specific policies and procedures)	SF 115 F–2070 F–2071 F–2073 F–3060 S–3091	X
☐	7.6	Wait for confirmation that any affected systems have been re-deployed and remove the system from the operational environment	(Operational environment with any previous external interfaces removed)		X
☐	7.7	Obtain final sign-off that decommissioning is complete	Go or No-Go Decision Form (Executive Sponsor signature)	F–1157	X

Appendix E. Transition of Legacy Systems

This appendix summarizes the process for transition of legacy systems from the NRC to a contractor for life-cycle management.

E.1 Systems Developed with SDLCM Methodology Guidance

This section addresses the case in which NRC or a contractor has developed an application system under the guidance of NRC's SDLCM Methodology. Under these conditions, when NRC transitions the application system to a different NRC organization or contractor for maintenance or enhancement, NRC will provide the system source code plus printed and electronic copies of all available system documentation, including at least the following:

- Project Definition and Analysis Document
- Software Engineering Notebooks
- Logical Design Document
- Physical Design Document
- Tactical Integration Plan
- Test Plan
- Integrated Education, Training, and Reference Materials
- Operational Support Guide
- User Guide
- On-Line Help System and Tutorials

The organization responsible for supporting the life-cycle management of the application system will maintain the system documentation to be consistent with changes in the operational system.

E.2 Systems Developed without SDLCM Methodology Guidance

This section addresses the case in which NRC or a contractor has developed an application system without the benefits that would have been derived by following the guidance of NRC's SDLCM Methodology. This situation may arise if the NRC wishes to transition a legacy system developed prior to NRC's introduction of the SDLCM Methodology or if a user who is unaware of the requirement to adhere to NRC's mandated system development methodology develops a new system.

Under those conditions, when NRC transitions the application system to a different NRC organization or contractor for maintenance or enhancement, NRC will provide the system source code plus printed and electronic copies of all available system documentation, including those listed if the preceding section, if possible.

A minimal subset of the system documentation is critical for the successful maintenance of any application. Without this minimal subset of system documentation, there is a high risk that the maintenance organization will be unable to satisfy requests for changes to the application system. If that subset is not available, NRC will fund the maintenance organization to develop the following minimal documentation at the time of transition:

- Project Definition and Analysis Document (portion):
  - Section 3, System Requirements Specification
- Software Engineering Notebook(s)
- Logical Design Document (portions):
  - Data Flow Diagrams
  - Data Dictionary
- Physical Design Document (portions):
  - Data Flow Diagrams
  - Data Dictionary
- Tactical Integration Plan (portions or optional subdocuments):
  - Solution Integration Plan
  - Other Systems Integration Plan
  - Network Integration Diagrams
- User Guide

Table E–1 summarizes the system documentation required for transition of an application system and references all of the products listed in Section E.1. For the convenience of the reader, the table also indicates the identifier of the SDLCM Methodology standard that defines the content of the each product. For potentially complex documents (that is, the PDAD, LDD, PDD, and TIP), the table lists sections (or optional subdocuments in the case of the TIP) separately.

The table also indicates the risk that maintenance will not be successful if the documentation is not provided.

- **High**. The documentation is *required* for successful maintenance. NRC must minimally provide the indicated products at the time of transition or must fund the maintenance organization to develop the products at that time.
- **Medium**. The documentation is *desirable*. If it is not provided, NRC will fund development of the documentation *when* is it required for specific maintenance or enhancement activities.
- **Low**. The documentation is *potentially useful* and will be provided to the maintenance organization if it is available. Although this documentation is essential when developing a new system, it is unlikely to be essential during maintenance. NRC will fund the creation of the documentation *if* it is needed for successful maintenance.

E.3 Alternative System Documentation

NRC's SDLCM Methodology provides an alternative standard to be used only for the case in which NRC or a contractor has developed an application system without the benefits that would have been derived by following the guidance of NRC's SDLCM Methodology. In such a situation, preparing the complete PDAD, LDD, and PDD after the application system has already been developed may not be useful. Portions of those products, although needed during the initial development of the application, may not be essential for system life-cycle management.

SDLCM Methodology Standard S–4151, As-Built System Documentation, consolidates the requirements and design of an existing system in one product to facilitate the maintenance and future enhancement of the application system. A documentation product built to comply with this standard contains all of the requirements and design information essential for successful maintenance and future enhancement activities. (See the standard for details of the content.) As the application system is modified through maintenance or enhancement, the maintenance organization will also maintain the as-built documentation to reflect the modified requirements and design.

A decision by NRC to adopt the As-Built System Documentation for transition of a legacy system that was not developed with SDLCM Methodology guidance simplifies the minimal subset of the system documentation that is critical for the successful maintenance of an application. Instead of the list prescribed in Section E.2, NRC may elect to fund the maintenance organization to develop the following minimal documentation at the time of transition:

- As-Built System Documentation
- Software Engineering Notebook(s)
- Tactical Integration Plan (portions or optional subdocuments):
  - Solution Integration Plan
  - Other Systems Integration Plan
  - Network Integration Diagrams
- User Guide

Table E–1. Documentation Required for System Transition

Project Product Name (specified by NRC's SDLCM Methodology)	Identifier	Risk if not provided		
		High	Med.	Low
Project Definition and Analysis Document	S–3051			
• (Section 3) System Requirements Specification	S–3051	✓		
• (Section 4) Data Requirements	S–3051		✓	
• 4.1 Entity List and Definitions	S–3051		✓	
• 4.2 Context Diagram	S–3162		✓	
• (Section 5) Assessment of Current System ***or*** Current System Assessment Document	S–3051 S–3052			✓
• (Section 7) Analysis of Alternatives	S–3051			✓
• (Section 6) System Operations Concept (SOC)	S–3051		✓	

(Continued on next page)

Table E–1. Documentation Required for System Transition (continued)

Project Product Name (specified by NRC's SDLCM Methodology)	Identifier	Risk if not provided		
		High	Med.	Low
Software Engineering Notebook	S–3091	✓		
Logical Design Document	S–3171			
• Data Models (Logical)	S–3151		✓	
• Process Models (Logical)	S–3161		✓	
• Context Diagram (Logical)	S–3162		✓	
• Data Flow Diagrams (Logical)	S–3163	✓		
• External Systems Interface Diagrams (Logical)	S–3164		✓	
• Data Dictionary (Logical)	S–3351	✓		
Physical Design Document	S–3172			
• Data Models (Physical)	S–3151		✓	
• Process Models (Physical)	S–3161		✓	
• Context Diagram (Physical)	S–3162		✓	
• Data Flow Diagrams (Physical)	S–3163	✓		
• External Systems Interface Diagrams (Physical)	S–3164		✓	
• Data Dictionary (Physical)	S–3351	✓		
Tactical Integration Plan (and related sub-documents)	S–5051			
• (Section 2) Rollout Plan	S–5051			✓
• (Section 3) Operations and Maintenance Plan	S–5051		✓	
• Conversion Plan	S–1054			✓
• Security Controls	S–1056			✓
• Products Installation and Integration Plan	S–5052		✓	
• Solution Integration Plan	S–5053	✓		
• Other Systems Integration Plan	S–5054	✓		
• Other Integration Issues Plan	S–5055		✓	
• Network Integration Diagrams	S–5056	✓		
• Installation Instructions	S–5252			✓
Test Plan	S–5151		✓	
Integrated Education, Training, and Reference Materials	S–7052			✓
Operational Support Guide	S–6151		✓	
User Guide	S–6051	✓		
On-Line Help System and Tutorials	S–6052		✓	

Appendix F. Glossary

Acceptance. An action by the customer (or an authorized representative) by which the customer assumes ownership of software products as partial or complete fulfillment of software requirements.

Acquire Support Resources. SDLCM Methodology Component 2, whose purpose is to obtain the necessary resources to ensure timely and effective progress on the project.

Activity. A unit of work that has well-defined entry and exit criteria. Activities can usually be broken into discrete steps.

Adapted unit. An existing unit that changes substantially (more than 25 percent of its content is changed, added, or deleted). Its origin is usually external to the project. (Contrast with *new unit*, *converted unit*, *transported unit*.)

Architecture. The organizational structure of a system or software CI, identifying its components, the component interfaces, and a concept of execution among the components.

Authentication. Code regeneration, quality assurance, code analysis, test and evaluation, verification and validation, and security aspects of configuration management. Configuration audits also fall under this heading. Authentication involves a series of inspections, tests, reviews and audits to ensure that each CI conforms with, and is accurately and completely described by, the CI's associated baseline documentation.

Baseline. A document or set of documents that describes a CI's characteristics and the verification required to demonstrate the achievement of those specified characteristics. After a baseline is established, the Nuclear Regulatory Commission (NRC) must approve all changes prior to their incorporation.

Build. A version of software that meets a specified subset of the requirements that the completed software will meet. (See also *release*.)

Certification. Written confirmation that a work product has been evaluated (for example, inspected or tested) and any defects found by that evaluation process have been satisfactorily resolved.

Component. 1. One of the seven structural divisions of the SDLCM Methodology. **2.** A module (See *module*.)

Computer software component (CSC). A logical subdivision of a computer software configuration item (CSCI). CSCs are a distinct part of a computer software configuration item. CSCs may be further decomposed into other CSCs and to yet smaller computer software units (CSUs).

Conceptual data model. One of the three data models; identifies groupings of data important to the business situation that the project addresses. (See also *logical data model*, *physical data model*.)

Conceptual process model. One of the two process models; also known as the scope-setting version of the logical process model. (See *context diagram*. See also *logical process model*.)

Conduct Initial Strategic Technology And Business Strategy Planning. Preliminary activities performed prior to applying the SDLCM Methodology

Configuration audit. The mechanism for verifying the system configuration.

Configuration identification. The application of a formal set of procedures to select CIs, determine the documentation required for each CI, affix unique identifiers to the documentation that defines the CI, release CIs and their associated documentation, and establish configuration baselines for each CI.

Configuration item (CI). System component (for example, hardware CI or software CI) that is developed or purchased, controlled, accepted, and maintained separately from other system components. In practice, it is a component that is convenient and sensible to document and control as an entity. (See *software configuration item*.)

Configuration management (CM). The application of a formal set of procedural concepts by which a uniform system of configuration identification, control, authentication, and status accounting is established and maintained for a system.

Configuration status accounting. The recording, monitoring, and reporting of all changes to established baselines for each CI. The purpose of configuration status accounting is to record and report all information needed to manage configuration items effectively. The objective of configuration status accounting is to ensure that all approved changes for a CI are reflected in associated baselines that describe the CI and that no baseline includes any changes that have not been approved.

Context diagram. A top-level data flow diagram that names the system to be developed or modified and that defines the bounds of a system in terms of the data it receives and generates. (See *conceptual process model, data flow diagram*.)

Converted unit. Existing unit that changes slightly (up to 25 percent of its content is changed, added, or deleted). Its origin is usually external to the project. (Contrast with *new unit, adapted unit, transported unit*.)

COTS (or **GOTS) software.** Commercial (or government), off-the-shelf software. COTS and GOTS software includes (1) unique components that must be delivered with the product to execute the operational software and (2) development tools that must be delivered with the product to support maintenance of the software.

Customer. The organization that procures software products for itself or another organization.

Cycle time. Elapsed calendar time (not effort) required to complete a given piece of work. For software efforts, cycle time usually refers to either (1) the full product engineering period (from initial receipt of product requirements through acceptance by the customer of the fully functional delivered product) or (2) a release cycle (from agreement on the requirements for the release through acceptance by the customer of the delivered product). The latter case, the release cycle, can pertain to new development releases as well as maintenance and enhancement releases. (See also *software life cycle.*)

Data Dictionary. A mechanism for defining the data elements identified in structured requirements analysis and design products (data flow diagrams, functional specifications, or the dictionary itself). It is a place for users, designers, programmers, and testers to determine what

constitutes data flows, data stores, and structure chart data couples; to look up unfamiliar terms; and to review data requirements.

Data flow diagram. A logical graphical representation of the data that are processed and generated during system operations.

Data model. (See *conceptual data model*, *logical data model*, and *physical data model*.)

Decommission The Solution. SDLCM Methodology Component 7, whose purpose is to inactivate and cease support of an application.

Define Initial Project Requirements. SDLCM Methodology Component 1, whose purpose is to identify an information management problem, clarify scope, identify functional and data requirements, analyze alternative solutions, select appropriate tools, establish a project action plan, and develop a support resource request.

Deploy the Solution. SDLCM Methodology Component 6, whose purpose is to implement and roll out the integrated solution.

Design. Those characteristics of a system or software CI that are selected by the software team in response to the requirements (see *requirements*). Some will match the requirements; others will be elaborations of requirements, such as definitions of all error messages in response to a requirement to display error messages; still others will be implementation related, such as decisions about what software units and logic to use to satisfy the requirements.

Design The Solution. SDLCM Methodology Component 3, whose purpose is to determine functional, data, and communications or network requirements, document user interface requirements, design the solution, and prepare integration and project plans.

Development. Production of a new product that leads to delivery of all initially planned functional capability. The new product may be built from newly created components, reused components (with or without adaptations), GOTS components, and COTS components. (Contrast with *enhancement, maintenance;* see also *software engineering*.)

Developmental configuration. The software and associated technical documentation that define the evolving configuration of a CI during development (that is, from the establishment of the allocated baseline to the establishment of the product baseline). All developmental configurations are under configuration control and describe the CI's design and implementation.

Deviation. A departure from a requirement, an alteration of an activity defined by a procedure, or a variation in the production of a product defined by a standard. (Contrast with *waiver*.)

Documentation. A collection of data, regardless of the medium (or media) on which it is recorded, that generally has permanence and can be read by humans or machines. The significance of this definition is that documents do not necessarily have to be separately bound entities; they may comprise data in several media (for example, plans, CASE tools, databases).

Engineer the Solution. SDLCM Methodology Component 4, whose purpose is to construct or acquire all modules of the system, perform tests, create the rollout strategy, and construct the training and education materials

Engineering change. A change to the currently approved configuration documentation of a configuration item at any point in the life cycle of the item.

Enhancement. A major addition or change in the functionality of an operational system; often includes many of the same activities as new development. (Contrast with *new development, maintenance;* see also *software engineering*.)

Firmware. The combination of a hardware device, computer instructions, and computer data that reside as read-only software on the hardware device.

GOTS software. (See *COTS software*.)

Inspection. A process whereby products are reviewed for correctness, completeness, quality, and compliance with requirements and standards. The process is carried out by one or more peers of the product's developer. (Contrast with *walkthrough*.)

Joint application design (JAD). A facilitated workshop technique that brings together principal stakeholders to solve a well-defined problem to produce a well-defined product or set of products.

Life cycle. (See *software life cycle.*)

Life-cycle model. A framework on which to map activities, methods, standards, procedures, tools, products, and services (for example, waterfall, spiral).

Life-cycle phase. A division of the software effort into non-overlapping time periods. Life-cycle phases are important reference points for the software manager. Multiple activities may be performed in a life-cycle phase; an activity may span multiple phases. (Contrast with *activity*.)

Logical data model. One of the three data models; the completed version that presents a fully attributed and normalized data model and identifies the relationships among all data entities. (See also *conceptual data model*, *physical data model*.)

Logical process model. One of the two process models; the completed version that provides all the details required to document and communicate to the business community the project team's understanding of the functional requirements of the application system. (See also *conceptual process model*.)

Maintenance. Implementation of problem fixes and minor enhancements to an operational product. (Contrast with *new development, enhancement;* see also *software engineering*.)

Measure. *n.* The extent, dimensions, capacity, etc., of anything, especially as determined by a standard; the act or process of measuring; the result of measuring.

Measure. *v.* To ascertain the quantity, mass, extent, or degree of something in terms of a standard unit or fixed amount, usually by means of an instrument or process; to compute the size of something from dimensional measurements; to estimate the extent, strength, worth, or character of something; to take measurements.

Measurement. The act or process of measuring something; the extent, quantity or size as determined by measuring; a system of measuring or of measures.

Method. A technique or approach, possibly supported by procedures and standards, that establishes a way of performing activities and arriving at a desired result.

Methodology. "A collection of methods, procedures, and standards that defines an integrated synthesis of engineering approaches to the development of a product." [Paulk]

Metric. *adj.* Designating the system of measurement based on the meter and gram. [*n.* Not properly a noun, but sometimes used loosely in other documents as a synonym for "measure."] (Contrast with *measure, measurement*.)

Metrics. *n.pl.* The theory or a system of measurement. [Sometimes used in other documents as a synonym for "measures" or "measurement."] (Contrast with *measure, measurement*.)

Milestone review. A process or meeting involving representatives of both the customer and the software team, during which project status, software products, and project issues are examined and discussed.

Module. A cohesive group of software units that performs a software function. (See also *component [2]*.)

New development. (See *development*.)

New technology. A technology that has not yet been proven to be effective in practice in a particular application area or domain. It may have been proven elsewhere, however. (See also *technology*.)

New unit. A software unit newly designed and developed for a new system. (Contrast with *adapted unit, converted unit, transported unit*.)

Non-developed item (NDI). A component that is not newly created by the software team. This includes reused components, COTS and GOTS components, and components developed by other government groups or contractor personnel.

Organization. "A unit within a company [for example, Computer Sciences Corporation] or other entity [for example, NRC] within which many projects are managed as a whole. All projects within an organization share a common top-level manager and common policies." [Paulk]

Phase. (See *life-cycle phase*.)

Physical data model. One of the three data models; the design model, which describes how data will be distributed to different processing nodes and structured to meet performance objectives in a specific physical implementation. (See also *conceptual data model*, *logical data model*.)

Policy. "A guiding principle, typically established by senior management, which is adopted by an organization or project to influence and determine decisions." [Paulk] (Contrast with *procedure, standard*.)

Procedure. A written description of the roles, responsibilities, and steps required for performing an activity or a subset of an activity. (Contrast with *policy, standard*.)

Process asset library (PAL). An organizational library, including a life-cycle methodology description, procedures and standards, guidebooks and handbooks, style guides, software profiles, key lessons, study results, tailored processes, examples of acceptable products, etc.

Process. "A sequence of steps performed for a given purpose; for example, the software engineering process." [IEEE 610.12]

Process Model. (See *conceptual process model, logical process model*.)

Product. (See *software product*.)

Project. (See *software project*. Contrast with *Technical Assignment Control*.)

Project process. A tailored version of the organization's standard process, defining and integrating specific life-cycle models, activities, methods, procedures, standards, and tools used to accomplish delivery of the project's required products and services. It is defined in the project's plan.

Prototype. An early experimental model of a system, system component, or system function that contains enough capabilities for it to be used to establish or refine requirements, or to validate design concepts.

Qualification testing. Testing performed to verify and demonstrate (often to the customer) that a software CI or a system meets its specified requirements.

Record. To *record* information means *to set down in a manner that can be retrieved and viewed.* The result may take many forms, including but not limited to hand-written notes, hard-copy or electronic documents, and data recorded in CASE and project management tools. Information to be delivered to the customer must be in the form, format, and medium agreed to with the customer. For information not to be delivered to the customer, the software manager selects the appropriate form, format, and medium.

Release. 1. A build that is delivered to the customer. (See *build*.) **2.** An action performed by the configuration management organization making a particular version of software available for a specific purpose (for example, for independent testing, or for operations).

Requirement. A characteristic that a system or software CI must possess in order to be acceptable to the customer.

Reused code. Code that has undergone no more than 25 percent change; that is, converted code and transported code. (See *converted unit, transported unit*.)

Service The Solution. SDLCM Methodology Component 6, whose purpose is to maintain, enhance, or modify an existing application or its support environment.

Software configuration item (CI). An aggregation of software that satisfies an end use function and is designated for separate configuration management by the customer or the maintenance team. Software CIs are selected on the basis of tradeoffs among software function, size, host or target computers, developer, support concept, plans for reuse, criticality, interface considerations, need to be separately documented and controlled, and other factors.

Software engineering environment. The facilities, hardware, software, firmware, procedures, and documentation needed to perform software engineering. Elements may include but are not limited to CASE tools, compilers, assemblers, linkers, loaders, operating systems, debuggers, simulators, emulators, documentation tools, and database management systems.

Software engineering process. An organized set of activities performed to translate user needs into software products.

Software engineering. A set of activities that results in software products. Software engineering includes new development, modification, reuse, reengineering, maintenance, or any other activities that result in software products.

Software life cycle. "The period of time that begins when a software product is conceived and ends when the software is no longer available for use." [IEEE 610.12] A life cycle is typically divided into life-cycle phases. (See *life-cycle phase*.)

Software process. (See *software engineering process*.)

Software product. Software or associated information created, modified, or incorporated to satisfy the software requirements. Examples include plans, requirements, design, code, databases, test information, and manuals.

Software product evaluation. Activities performed by the software team to ensure that in-process and final software products meet criteria established for those products. (Contrast with *software quality assurance*.)

Software project. An undertaking requiring a concerted effort, which is focused on developing or maintaining a specific software product. Typically a software project has its own funding, cost accounting, and delivery schedule. That is, the project is the work to be done and the personnel assigned to perform the work; it is not the same as the product (for example, system) that they are to produce.

Software quality assurance (SQA). Activities performed by independent QA personnel to (1) ensure that each activity described in the software plan is performed in accordance with the software plan, and (2) ensure that each software product required by the organization or by software requirements exists and has undergone software product evaluations, testing, and corrective action as required. (Contrast with *software product evaluation*.)

Software system. A system consisting solely of software and possibly the computer equipment on which the software operates.

Software team. The group that engineers software products (including new development, modification, reuse, reengineering, maintenance, or any other activity that results in software products).

Software test environment. The facilities, hardware, software, firmware, procedures, and documentation needed to perform qualification, and possibly other, testing of software. Elements may include but are not limited to simulators, code analyzers, test plan generators, and path analyzers, and may also include elements used in the software engineering environment.

Software transition. The set of activities that enables responsibility for software engineering to pass from one organization, usually the organization that performs initial software development, to another, usually the organization that will perform sustaining engineering.

Software unit. Smallest physical element of software processed by source code translators, compilers, and assemblers.

Standard. Written criteria used to develop and evaluate a product or to provide and evaluate a service. (Contrast with *policy, procedure*.)

Sustaining engineering. Maintenance and enhancement of an operational product. (See *maintenance*, *enhancement*.)

System. Operational entity composed of a set of interrelated and cohesive configuration items (CIs). (See *configuration item (CI)*.)

Task Assignment Control (TAC). A contractual mechanism for initiating and funding a project, multiple projects, or a portion of a project. (Contrast with *project*.)

Technology change management. "... [I]dentifying, selecting, and evaluating new technologies, and incorporating effective technologies into the organization. The objective is to improve software quality, increase productivity, and decrease cycle time for product engineering." [Paulk]

Technology infusion. (Used synonymously with technology transfer; see *technology transfer*.)

Technology management. (See *technology change management*.)

Technology transfer. The successful importing or exporting of technology from lab to practice or from practice to practice.

Technology utilization. The application and integration of appropriate technologies in software efforts.

Timebox. A subdivision of a release created to achieve a unit of production that is manageable in size, complexity, staffing, and duration. It is a planning construct that controls functionality delivered by fixing resources and time.

Transported unit. Existing unit that is used verbatim (except for possible changes to the development history for traceability). Its origin is usually external to the project. (Contrast with *new unit, adapted unit, converted unit*.)

Unit. (See *software unit*.)

Validation. "The process of evaluating software during or at the end of the software development process to determine whether it satisfies specified requirements." [IEEE 610.12] (Contrast with *verification*.)

Verification. "The process of evaluating software to determine whether the products of a given development phase [or activity] satisfy the conditions imposed at the start of that phase [or activity]." [IEEE 610.12] (Contrast with *validation*.)

Version. An identified, documented, and controlled collection of software. Any changes to a version of software and its associated documentation require configuration management actions.

Waiver. The elimination of a specific requirement to perform an activity or to produce a product. (Contrast with *deviation*.)

Walkthrough. Detailed technical presentation of a limited aspect or portion of a product. Such presentations are usually made by the product's developer. (Contrast with *inspection*.)

Acronyms

BPR	business process reengineering
CCB	configuration control board
CDR	Critical Design Review
CI	configuration item
CIO	Chief Information Officer
CM	configuration management
CMO	Configuration Management Office
CMP	Configuration Management Plan
COTS	commercial off-the-shelf
CP	Change Proposal
CPIC	Capital Planning and Investment Control
CSAD	Current System Assessment Document
CSC	Computer Sciences Corporation
CSC	computer software component
CSCI	computer software configuration item
CSU	computer software unit
DFD	data flow diagram
DM	data management
FCA	Functional Configuration Audit
GILS	Government Information Locator System
GOTS	government off-the-shelf
GSA	General Services Administration
IRM	information resources management
IT	information technology
JAD	joint application design
LDD	Logical Design Document
NDI	non-developed item
NRC	Nuclear Regulatory Commission
OCIO	Office of Chief Information Officer
OIRM	Office of Information Resources Management

PAL	process asset library
PAP	Project Action Plan
PCA	Physical Configuration Audit
PDAD	Project Definition and Analysis Document
PDD	Physical Design Document
PFD	process flow diagram
PR	Problem Report
QA	quality assurance
QAP	Quality Assurance Plan
RM	records management
SDIB	Systems Development and Integration Branch
SDLCM	System Development and Life-Cycle Management
SDLCMM	System Development and Life-Cycle Management Methodology
SEN	software engineering notebook
SME	subject matter expert
SOC	System Operations Concept
SOW	Statement of Work
SQA	software quality assurance
SRS	System Requirements Specification
SSR	Software Specification Review
TAC	Task Assignment Control
TBD	to be determined
TIP	Tactical Integration Plan
WBS	work breakdown structure

References

[IEEE 610.12] ANSI/IEEE-STD-610.12-1990, "IEEE Standard Glossary of Software Engineering Terminology," February 1991.

[ISO/SEC 12207] ISO/SEC 12207, "Software Life-Cycle Processes."

[MIL-STD-498] MIL-STD-498, "Software Development and Documentation.

[NRC MD2.5] NRC Management Directive 2.5, "Application Systems Life-Cycle Management," Draft, January 1997.

[NRC SH1.1] NRC OIRM/SDIB, *SDLCM Methodology Handbook*, Version 1.1, November 1996.

[NRC SH2.0] NRC OCIO/ADD, *SDLCM Methodology Handbook*, Version 2.0, December 1997.

[Paulk] Paulk, M, et al., *Key Practices of the Capability Maturity Model, Version 1.1,* Software Engineering Institute, Carnegie Mellon University, CMU/SEI-93-TR-25, February 1993.