System Development Life Cycle Guide

Version 1.1 ● 30 MAY 2008

Version History

This and other Framework Extension tools are available on Framework Web site.

Release Date	Description
30-May-2008	Version 1.1 released. Modified all references to “Project Plan and related plans” to “Project Plan” in order to align with Framework 2.0 and Change Request 38.
25-Sep-2007	Version 1.0 – System Development Life Cycle Guide released.

Contents

Introduction

The System Development Life Cycle (SDLC) Guide (Guide) provides direction on using and tailoring the Texas Project Delivery Framework (Framework) SDLC Extension toolset. In addition, the Guide addresses integration of the SDLC with other processes that may be used during a project or system life cycle from the conception of ideas to the retirement of a system.

Framework Extensions provide a standard set of guidance and tools that extend use of the Framework for various types of technology projects and environments. The SDLC Extension, when specifically referenced, is a separate and distinct extension of the base Framework for developing a system and/or system components. System development is a process of defining the hardware and software architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. It also includes requirements definition, design, testing, and implementation. This process may require customizing or tailoring the SDLC, based on project and product requirements.

The content of this Guide is based on the following standards:

- IEEE (Institute of Electrical and Electronics Engineers)/ EIA (Electronic Industries Alliance) Standard 12207.0 1995 Industry Implementation of International Standard ISO (International Standard Organization)/IEC (International Electrotechnical Commission) 12207: 1995 Standard for Information Technology – Software Life Cycle Processes
- IEEE Standard 15288 - 2004 (Adoption of ISO/IEC 15288:2002), Systems Engineering – System Life Cycle Processes

Use of the System Development Life Cycle Guide

Since the SDLC Extension extends use of the Framework, the assumption is that users of the Guide have an understanding of the Framework guidance and tools. Refer to the Framework Handbook for additional information.

This Guide can be used as a resource to aid practitioners in customizing or tailoring the system development process. The system development process is applicable to engineering or reengineering of systems or any portion thereof, regardless of size, complexity, or technology. System development includes systems containing hardware, software, firmware, humans, data, services, and processes.

The information provided in this Guide for each SDLC activity can be used as an aid in determining which activities and tasks are appropriate for development of a system or system component for a specific project. In addition, the Guide provides assistance in determining which SDLC Extension deliverables are appropriate based on risk associated with the project or technical solution.

To effectively tailor and execute the SDLC, use the Guide to focus on the impact of the system and system components produced. Development of a quality system and system components positively impacts the quality, cost, schedule, and scope of a project.

Note: The Guide assumes that project planning information contained in the Project Plan will be updated appropriately during execution of SDLC activities; therefore the plan is not included in the outputs for SDLC activities.

Section 1. System Life Cycle Processes

1.1 INTRODUCTION

System life cycle processes are used by organizations and projects to manage and perform the stages of a system's life cycle (e.g., concept, development, production, retirement). System life cycle processes are used to acquire, supply, develop, operate, and maintain systems and system components, spanning the life of the system or component from conception and definition of requirements to the termination of use.

Figure 1 illustrates system life cycle processes, which may be concurrently and iteratively executed to deliver a system and/or system components. For example, concurrent use of system life cycle processes can occur when Technical Processes are executed to build a system or system component while Project Processes are executed in order to plan, execute, and monitor those Technical Processes and other aspects of the project.

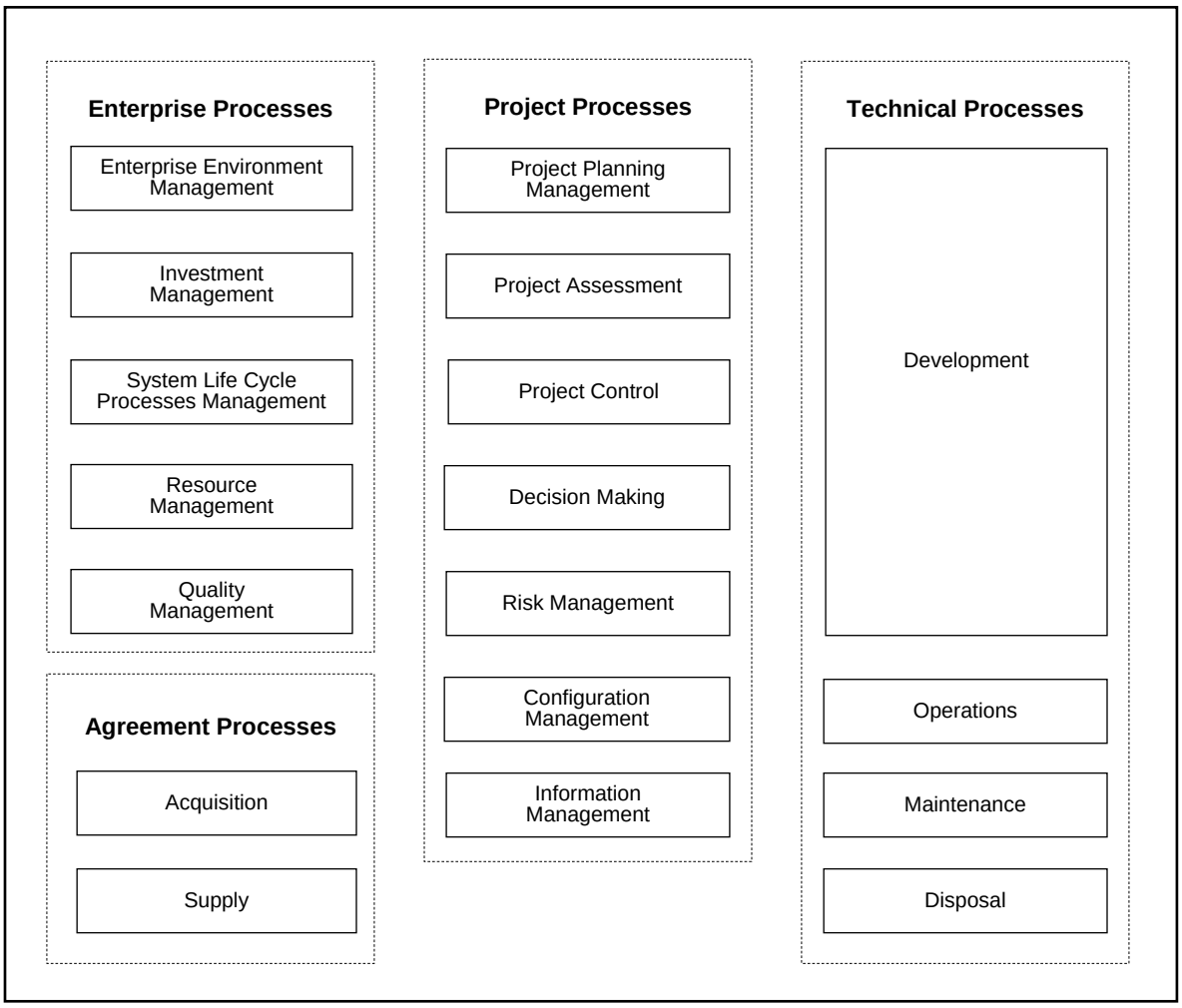


Figure 1. Illustration of system life cycle processes

In addition to concurrent use of processes within a single project, concurrent use of processes may occur among multiple projects when technology activities for building various components of a system are performed at the same time under different project responsibility, or when building a system that is required to interface with another system. Figure 2 illustrates concurrent execution of Enterprise, Project, Agreement, and Technical Processes among two projects or areas of responsibility.

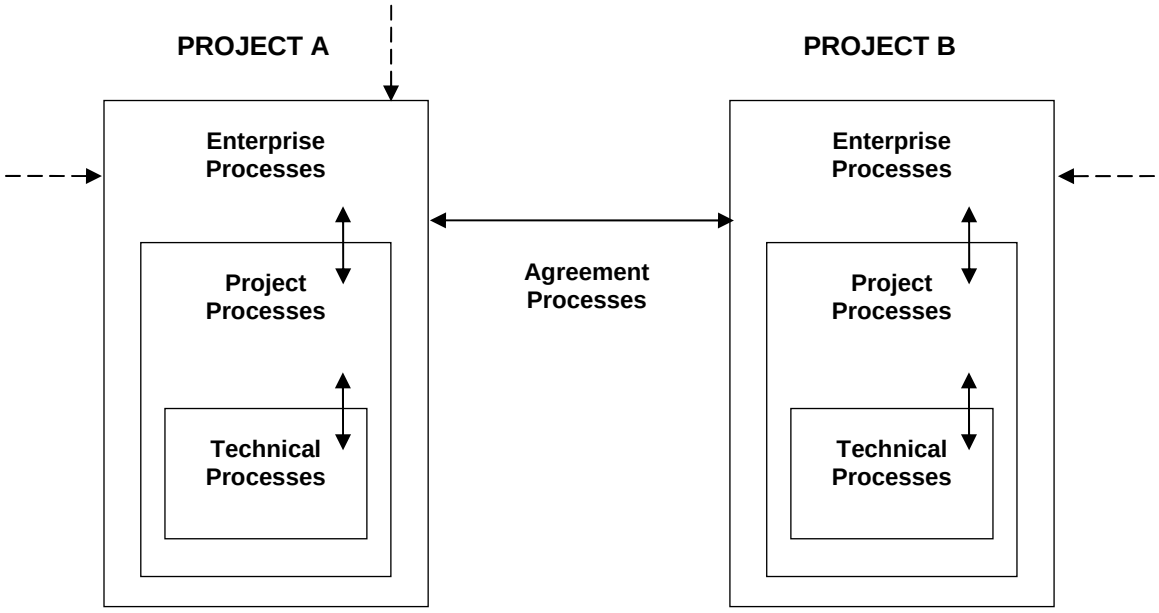


Figure 2. Illustration of concurrent execution of system life cycle processes between two projects

1.2 SYSTEM LIFE CYCLE PROCESSES AND THE ORGANIZATION

As illustrated in Figure 1, the Development Process, a subset of Technical Processes, is intended to be integrated with the other system life cycle processes. This Guide provides a model for using the Framework SDLC Extension toolset for the Development Process. In addition to defining the Development Process and other Technical Processes, it is an agency's or organization's responsibility to define other system life cycle processes such as: Enterprise, Agreement, and Project Processes. For example, agencies or organizations are required to define project management practices for use in technology projects. In addition, these practices must incorporate the use of the Framework toolset for major information resource projects.

Section 2. Development Process

2.1 INTRODUCTION

As illustrated in Figure 1, the Development Process is a subset of the Technical Processes within a system life cycle process. Figure 3 illustrates the subset of Technical Processes that constitute the Development Process. The Development Process is used to define the requirements for a system or system component and to transform the requirements into an effective product that provides the required services. The Development Process includes the activities for requirements analysis, design, coding, integration, testing, installation, and acceptance of systems and system components.

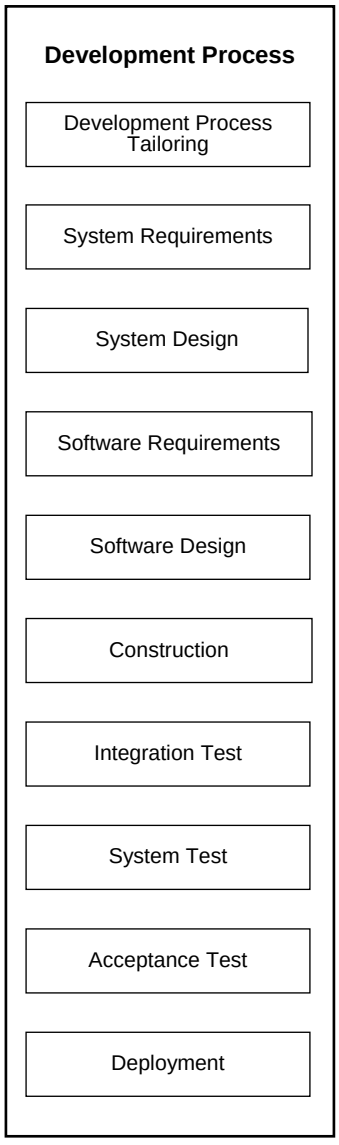


Figure 3. Illustration of the Development Process—a subset of Technical Processes

2.2 SYSTEM DEVELOPMENT LIFE CYCLE MODELS

A system development process is a structure imposed on the development of a system product. There are several models for such processes, each describing approaches to a variety of activities and/or tasks that take place during the process.

The Framework SDLC Extension is based on the V-model, illustrated in Figure 4. **Basing the SDLC Extension on the V-model does not preclude application and use of various types of life cycle models, including hybrids.** Examples of other system development life cycle models, including the Incremental model, Rapid Application Development model, and Agile model, are described in the Appendices.

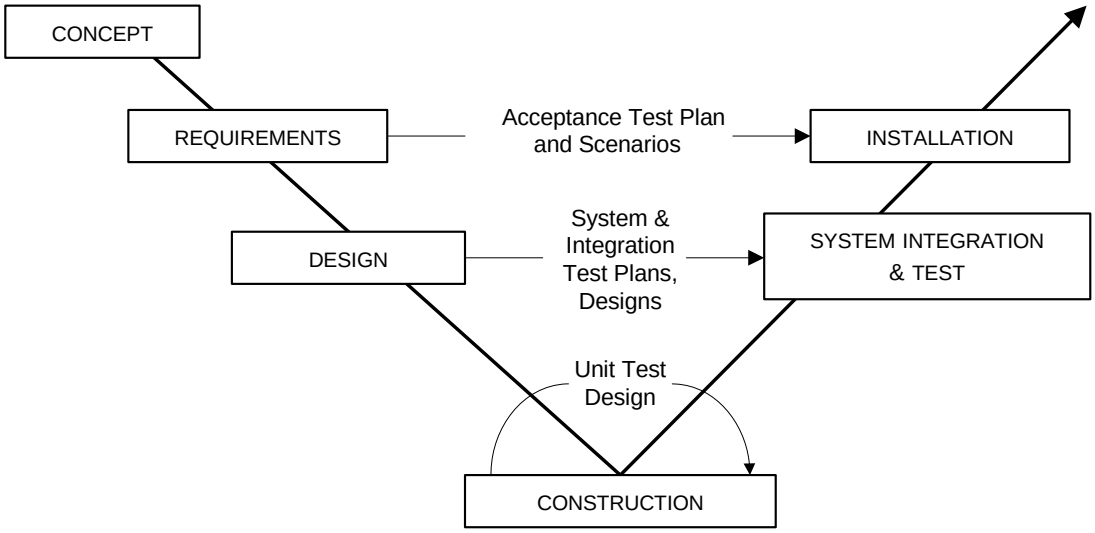


Figure 4. Illustration of the V-model system development process

This V-shaped life cycle is a sequential path of execution of technology activities that emphasizes planning and execution of test phases. Planning for each phase of testing is initiated early in the life cycle, before any coding is done.

As in many other life cycle models, the V-model begins with initial elicitation of system requirements in order to conceptualize the system. Once the system concept is formed, requirements are further elicited, in order to specify detailed system requirements. These system requirements are then allocated to system components (e.g., software, hardware, firmware, and people) and the system is designed. Once the system is designed, the detailed software requirements are elicited. The software requirements are then used to design, construct, and test appropriate components of the system. Eventually, all of the system components are integrated and tested. When evidence exists that the resulting system meets requirements, the system and its components are accepted.

The V-model integrates planning and execution of testing throughout the life cycle. During requirements elicitation and analysis, acceptance test planning is initiated, including development of acceptance test scenarios. The plans and scenarios for acceptance testing are executed prior to deploying the system.

During design activities, system and integration test planning and scenarios are initiated. These test plans and scenarios are executed after units of software are coded and unit tested.

Section 3. System Development Life Cycle Activities

The following sections specify the purpose, inputs, outputs, entry and exit criteria, and the sequence of activities and tasks to be initiated and completed within the SDLC.

- **Entry criteria** include the conditions that must exist before an activity can begin.
- **Inputs** for each activity include the items that may be needed to perform the tasks within the activity.
- **Exit criteria** include the conditions that must exist before an activity can end. Specified exit criteria must be met if the activity and task that causes or enables the criteria to be met is included within the SDLC Model.
- **Outputs** include the items that may be produced as a result of performing the activity.

The Guide assumes that project planning information contained in the Project Plan will be updated appropriately during execution of SDLC activities; therefore the plan is not included in the outputs for SDLC activities. The SDLC tool associated with each task is specified in the SDLC Tool column in the Activity Tasks table. Other tools may be used to complete SDLC activities.

Some activities require tasks that involve planning future SDLC activities and tasks. For example, the System Requirements and System Design activities include tasks to initiate and/or construct Test Plan and Deployment Plan content. The Test Plan and Deployment Plan are actually completed, reviewed, and approved in activities and tasks that occur much later in the SDLC.

To aid in assessing the value of including a particular Development Process activity or task based on risks associated with the project or technology solution, Tailoring Considerations are included for each activity.

Note that the first activity within the SDLC is Development Process Tailoring. The purpose of this activity and the tasks within the activity is to provide guidance in tailoring the remainder of the SDLC activities and tasks, based on their usefulness in enabling the project delivery and system development practitioners to deliver an effective product.

Flow charts for each activity described in this section are included in the Appendices.

3.1 DEVELOPMENT PROCESS TAILORING

Purpose

To identify the System Development Life Cycle Model and select and map onto the Life Cycle Model the activities and tasks to be included in the Development Process based on the scope, magnitude, and complexity of the project.

Tailoring Considerations

When developing plans for conducting all activities of the Development Process, consider standards, methods, tools, actions, and responsibilities associated with development activities, including the qualification of requirements. Use project planning, risk, and requirements information, and the Tailoring Considerations for each SDLC activity to aid in development of these plans.

Consider which unfavorable circumstances could occur if a particular course of action is taken within the SDLC. Determine how risks can be avoided or mitigated by the selection of particular activities and tasks to include within the SDLC Model. For example, consider whether risk would be increased if appropriate stakeholder participation in requirements elicitation activities is omitted, or by omitting documentation of system and/or software requirements. If appropriate stakeholders do not participate in the requirements elicitation activities or requirements are not documented, requirements for the project may not be correct, or fully defined or understood by the development team. Incorrect and unclear requirements will impact the quality, cost, schedule, and scope of a project.

Entry Criteria

- Project Charter approved

Inputs

- Business Case baseline
- Statewide Impact Analysis (SIA) baseline
- Project Charter baseline
- Project Plan draft
- Organizational methods, standards, and best practices
- Existing system and software baselines
- Existing SDLC deliverables, if applicable (Test Plans and Scenarios, requirements documentation)

Activity Tasks

Task Number	Task	SDLC Tool
3.1.1	Perform SDLC tailoring analysis. Select and map activities and tasks.	• SDLC Guide
3.1.2	Document SDLC Model appropriate to the scope, magnitude, and complexity of the project (includes activities, tasks, and deliverables).	
3.1.3	Integrate SDLC activities, tasks, and deliverables into project planning information.	

Exit Criteria

- SDLC Model tailored, as appropriate, for the scope, magnitude, and complexity of the project
- SDLC activities and tasks documented and integrated into the project planning information

Outputs

- SDLC Model

3.2 SYSTEM REQUIREMENTS

Purpose

To establish a common understanding of the system requirements between the project team and stakeholders. System requirements are elicited from stakeholders, analyzed, and presented in a System Requirements Specification (SyRS). The SyRS describes functions and capabilities of the system; business, organizational, and user requirements; safety, security, human-factors, interface, operations, and maintenance requirements; design constraints and qualification requirements.

Tailoring Considerations

Consider that eliciting and documenting system requirements facilitates a dialogue between the project team and the stakeholders. This dialogue helps the project team to understand what is required of the intended product and reduces the likelihood of there being missing or inaccurate requirements. When requirements are translated into a system design and implemented, the missing and inaccurate requirements may negatively impact the quality of the resulting system. When problems with quality occur, requests to change the system may be initiated and change requests impact cost, schedule, and scope of a project.

The SyRS is used as the basis for the design and implementation of the system. Development of the SyRS is performed iteratively with the activities within the System Design process. Refer to the System Requirements Specification Instructions for the fundamental characteristics of requirements.

It is imperative that the SyRS is verified by the stakeholders to ensure that the system requirements are correct and complete and that they are aligned with the concept or solution documented in the Business Case.

Another important task in documenting system requirements is the initiation of the Requirements Traceability Matrix (RTM). The RTM facilitates backward and forward traceability of all requirements. Use of traceability confirms that all requirements have been accounted for within the entire SDLC. Traceability also aids in identifying requirements that are either missing from, or in addition to, the original requirements. It ensures that the product delivered satisfies the agreement between the project team and the stakeholders.

In addition, it is important to consider that tasks within this activity include acceptance test and deployment planning. As discussed in the System Development Life Cycle Models section, planning and execution of testing is integrated throughout the life cycle. During requirements elicitation and analysis, acceptance test planning should be initiated, including development of acceptance test scenarios. Initiation of test and deployment planning within requirements activities, and iterative refinement of those plans throughout other life cycle activities, results in alignment and traceability among requirements, the actual products, and the plans. It also results in enabling successful execution of test and deployment plans later in the life cycle.

When requirements dictate that only a software system component will be developed and not an entire system, the System Requirements activity may be excluded from the project's SDLC Model.

Entry Criteria

- Project Plan approved and baselined

Inputs

- Business Case baseline
- Project Plan baseline
- Organizational methods, standards, and best practices
- Existing system and software baselines
- Existing SDLC deliverables, if applicable (Test Plans and Scenarios, requirements documentation)

Activity Tasks

Task Number	Task	SDLC Tool
3.2.1	Elicit system requirements.	
3.2.2	Construct System Requirements Specification (SyRS).	• System Requirements Specification
3.2.3	Initiate Requirements Traceability Matrix (RTM).	• Requirements Traceability Matrix (RTM)
3.2.4	Conduct review of SyRS. Requirements should be reviewed prior to initiating system design.	• System Development Review Guidelines • Requirements Verification Checklist
3.2.5	Obtain SyRS approval.	
3.2.6	Begin construction of Test Plan. Address Acceptance Test Phase information.	• Test Plan
3.2.7	Begin Construction of Acceptance Test Scenarios.	• Test Scenario
3.2.8	Update RTM. Address traceability from requirements to Test Scenarios.	• Requirements Traceability Matrix (RTM)
3.2.9	Begin construction of Deployment Plan	• Deployment Plan
3.2.10	Capture baselines and/or place under configuration management: • SyRS baseline • RTM baseline • Test Plan draft • Acceptance Test Scenarios draft • Deployment Plan draft • Other deliverables, if applicable	

Exit Criteria

- SyRS is approved and baselined

Outputs

- SyRS baseline
- RTM baseline
- Test Plan draft
- Acceptance Test Scenarios draft
- Deployment Plan draft
- Other deliverables, if applicable

3.3 SYSTEM DESIGN

Purpose

To determine and document in sufficient detail how the system is to be constructed. The system design must meet system requirements and standards. The system design is the top-level system architecture and identifies all components of hardware, software, and manual operations. In addition, it identifies system-wide design decisions, concept of execution, interface design, and requirements traceability to design components.

Tailoring Considerations

Consider that determining and documenting the system design can reduce project risk by reducing uncertainty in the implementation of the system. Documentation of the system design contributes to the success of the system by establishing and communicating how the properties of the system requirements will be transitioned into a design. This documentation enables expectations for all aspects of the system's features and performance to be contrasted with the design in order to identify and resolve unallocated requirements and potential design flaws. Identification and resolution of unallocated requirements, design flaws, and problems positively impact the quality of and customer satisfaction with the implemented system. In addition, flaws and problems corrected early in system development have less of an impact on a project's schedule and budget.

In many cases, system design may be an iterative activity. Final identification of all components and allocation of all requirements may be completed or finalized in iterations other than the first. Refer to the System Design Description (SyDD) Instructions for content of the SyDD.

In addition, it is important to consider that tasks within this activity include test and deployment planning. As discussed in the System Development Life Cycle Models section, planning and execution of testing is integrated throughout the life cycle. Initiation of test and deployment planning occurs within the System Requirements activity and iterative refinement of those plans occurs throughout subsequent life cycle activities. Initiation and refinement of test scenarios also occurs, as necessary. This progression results in alignment and traceability among requirements, the actual products, and the plans. It also results in enabling successful execution of test and deployment plans later in the life cycle.

Alignment with Enterprise Processes, shown in Figure 1, should be considered during the System Design activity, including feasibility of hardware and software items fulfilling their allocated requirements, and feasibility of operation and maintenance.

When business requirements dictate that only a software system component will be developed and not an entire system, the System Design activity may be excluded from the project's SDLC Model.

Entry Criteria

- Project Plan approved and baselined
- SyRS approved and baselined

Inputs

- Business Case baseline
- Project Plan baseline
- SyRS baseline

- RTM baseline
- Test Plan draft
- Acceptance Test Scenarios draft
- Deployment Plan draft
- Organizational methodologies, standards, and best practices
- Existing system and software baselines
- Existing SDLC deliverables, if applicable (Test Plans and Scenarios, requirements documentation)

Activity Tasks

Task Number	Task	SDLC Tool
3.3.1	Allocate functional system requirements to system components (software, hardware, people).	• Requirements Traceability Matrix (RTM)
3.3.2	Develop system design.	
3.3.3	Construct System Design Description (SyDD).	• System Design Description
3.3.4	Update RTM.	• Requirements Traceability Matrix (RTM)
3.3.5	Conduct review of SyDD. SyDD review should be conducted prior to initiating elicitation of software requirements.	• System Development Review Guidelines • System Design Verification Checklist
3.3.6	Obtain SyDD approval.	
3.3.7	Continue construction of Test Plan. Address the following information within the Test Plan: • Test Plan Overview • Test Methodology • Integration Test Phase • System Test Phase • Acceptance Test Phase • Test Schedule • Test Monitoring and Reporting	• Test Plan
3.3.8	Begin/continue construction of Test Scenarios. Include traceability to requirements. Address the following scenarios: • Integration Test • System Test • Acceptance Test	• Test Scenario
3.3.9	Update RTM. Address traceability from requirements to Test Scenarios.	• Requirements Traceability Matrix (RTM)
3.3.10	Continue construction of Deployment Plan.	• Deployment Plan
3.3.11	Capture baselines and/or place under configuration management: • SyDD baseline • RTM baseline • Test Plan draft • Test Scenario drafts • Deployment Plan draft • Other deliverables, if applicable	

Exit Criteria

- System Requirements allocated and documented in RTM
- SyDD and RTM approved and baselined

Outputs

- SyDD baseline
- RTM baseline
- Test Plan draft
- Test Scenario drafts
- Deployment Plan draft
- Other deliverables, if applicable

3.4 SOFTWARE REQUIREMENTS

Purpose

To establish a common understanding of the software requirements between the project team and stakeholders. Requirements allocated to software items are elicited from stakeholders, analyzed, and presented in a Software Requirements Specification (SRS).

Tailoring Considerations

Consider that eliciting and documenting software requirements facilitate a dialogue between the project team and the stakeholders. This dialogue reduces the likelihood of there being missing or inaccurate requirements. When requirements are translated into a software design and implemented, the missing and inaccurate requirements may negatively impact the quality of the resulting system. When problems with quality occur, requests to change the system may be initiated, and change requests impact cost, schedule, and scope of a project.

The SRS is used as the basis for the design and implementation of the software. Refer to the Software Requirements Specification Instructions for the fundamental characteristics of requirements.

It is imperative that the SRS is verified by the stakeholders to ensure that the software requirements are correct and complete and that they are aligned with the selected solution and the system requirements and design.

Another important task in documenting software requirements is updating the Requirements Traceability Matrix (RTM). The RTM facilitates backward and forward traceability of all requirements. Use of traceability confirms that all requirements have been accounted for within the entire SDLC. Traceability also aids in identifying requirements that are either missing from, or in addition to, the original requirements. It ensures that the product delivered satisfies the agreement between the project team and the stakeholders and that all requirements allocated to a particular software item are addressed.

In addition, it is important to consider that tasks within this activity include test and deployment planning. As discussed in the System Development Life Cycle Models section, planning and execution of testing is integrated throughout the life cycle. During software requirements elicitation and analysis, the test plan and scenarios should be updated. Initiation of test and deployment planning occurs within the System Requirements activity and iterative refinement of those plans and scenarios occurs throughout subsequent life cycle activities. This progression results in alignment and traceability among requirements, the actual products, and the plans. It also results in enabling successful execution of test and deployment plans later in the life cycle.

When business requirements dictate that only a software system component will be developed and not an entire system, the SDLC Model for the project may begin with the activity of documenting software requirements. When the SDLC Model is tailored in this manner, the RTM is initiated within this activity.

Entry Criteria

- Project Plan approved and baselined
- SyRS approved and baselined, if applicable
- SyDD approved and baselined, if applicable
- System Requirements allocated and documented in RTM, if applicable

Inputs

- Business Case baseline
- Project Plan baseline
- SyRS baseline
- RTM baseline
- SyDD baseline
- Test Plan draft
- Test Scenario drafts
- Organizational methodologies, standards, and best practices
- Existing system and software baselines
- Existing SDLC deliverables, if applicable (Test Plans and Scenarios, requirements documentation)

Activity Tasks

Task Number	Task	SDLC Tool
3.4.1	Elicit software requirements.	
3.4.2	Construct Software Requirements Specification (SRS).	• Software Requirements Specification
3.4.3	Update RTM.	• Requirements Traceability Matrix (RTM)
3.4.4	Conduct review of SRS. Requirements should be reviewed prior to initiating software design.	• System Development Review Guidelines • Requirements Verification Checklist
3.4.5	Obtain SRS approval.	
3.4.6	Continue construction of Acceptance Test Phase information within the Test Plan.	• Test Plan
3.4.7	Continue construction of Test Scenarios. Include traceability to requirements.	• Test Scenario
3.4.8	Update RTM. Address traceability from requirements to Test Scenarios.	• Requirements Traceability Matrix (RTM)
3.4.9	Continue construction of Deployment Plan.	• Deployment Plan
3.4.10	Capture baselines and/or place under configuration management: • SRS baseline • RTM baseline • Test Plan draft • Test Scenario drafts • Deployment Plan draft • Other deliverables, if applicable	

Exit Criteria

- SRS approved and baselined
- RTM baselined

Outputs

- SRS baseline
- RTM baseline
- Test Plan draft
- Test Scenario drafts
- Deployment Plan draft
- Other deliverables, if applicable

3.5 SOFTWARE DESIGN

Purpose

To determine and document in sufficient detail how the software is to be constructed. The software design must meet software requirements and standards. The software design is the top-level software structure and identifies all software items. This information is further refined to facilitate details of the software architecture and design. In addition, it identifies requirements traceability to design features and decisions about the software's behavioral design and other decisions affecting the selection and design of software items, when appropriate.

Tailoring Considerations

Consider that providing documentation of the software design can reduce project risk by reducing uncertainty in the implementation of the software. Documentation of the software design contributes to the success of the software and overall system by establishing and communicating how the properties of the software requirements will be transitioned into a design. This documentation enables expectations for all aspects of the software's features and performance to be contrasted with the design in order to identify and resolve potential design flaws. Identification and resolution of design flaws and problems positively impact the quality of and customer satisfaction with the implemented system. In addition, flaws and problems corrected early in software development have less of an impact on a project's schedule and budget.

It is important to consider that tasks within this activity include test and deployment planning. As discussed in the System Development Life Cycle Models section, planning and execution of testing is integrated throughout the life cycle. Initiation of test and deployment planning occurs within the System Requirements activity and iterative refinement of those plans occurs throughout subsequent life cycle activities. Initiation and refinement of test scenarios also occurs, as necessary. This progression results in alignment and traceability among requirements, the actual products, and the plans. It also results in enabling successful execution of test and deployment plans later in the life cycle.

Alignment with Enterprise Processes, shown in Figure 1, should be considered during Software Design, including feasibility of design decisions and feasibility of operation and maintenance. Refer to the Functional Software Design Description Instructions or Object-oriented Software Design Description Instructions for content of the Software Design Description (SDD).

Entry Criteria

- Project Plan approved and baselined
- SyRS approved and baselined, if applicable
- SyDD approved and baselined, if applicable
- System Requirements allocated and documented in RTM, if applicable
- SRS approved and baselined

Inputs

- Business Case baseline
- Project Plan baseline
- SyRS approved and baselined, if applicable
- SyDD approved and baselined, if applicable

- RTM baseline
- SRS baseline
- Test Plan draft
- Test Scenario drafts
- Deployment Plan draft
- Organizational methodologies, standards, and best practices
- Existing system and software baselines
- Existing SDLC deliverables, if applicable (Test Plans and Scenarios, requirements documentation)

Activity Tasks

Task Number	Task	SDLC Tool
3.5.1	Develop software design.	
3.5.2	Construct Software Design Description (SDD).	• Functional Software Design Description • Object-oriented Software Design Description
3.5.3	Update RTM.	• Requirements Traceability Matrix (RTM)
3.5.4	Conduct review of SDD. SDD review should be conducted prior to initiating coding.	• System Development Review Guidelines • Functional Software Design Verification Checklist • Object-oriented Software Design Verification Checklist
3.5.5	Obtain SDD approval.	
3.5.6	Continue construction of Test Plan. Address the following information within the Test Plan: • Test Plan Overview • Test Methodology • Integration Test Phase • System Test Phase • Acceptance Test Phase • Test Schedule • Test Monitoring and Reporting	• Test Plan
3.5.7	Continue construction of Test Scenarios. Include traceability to requirements. Address the following scenarios: • Integration Test • System Test • Acceptance Test	• Test Scenario
3.5.8	Update RTM.	• Requirements Traceability Matrix (RTM)
3.5.9	Continue construction of Deployment Plan.	• Deployment Plan
3.5.10	Capture baselines and/or place under configuration management: • SDD baseline • RTM baseline • Test Plan draft • Test Scenario drafts • Deployment Plan draft • Other deliverables, if applicable	

Exit Criteria

- SDD approved and baselined
- RTM baselined

Outputs

- SDD baseline
- RTM baseline
- Test Plan draft
- Test Scenario drafts
- Deployment Plan draft
- Other deliverables, if applicable

3.6 CONSTRUCTION

Purpose

To construct and unit test the software product described by the Software Design Description.

Tailoring Considerations

Consider that this activity consists of performing detailed design, coding, and testing for each software unit to satisfy the requirements and design described in the Software Requirements Specification and Software Design Description. In addition, consider that this activity also contains tasks to perform detailed design and coding of databases and interfaces between software units and items, and planning and execution of unit tests.

It is important to consider that tasks within this activity include test planning. As discussed in the System Development Life Cycle Models section, planning and execution of testing is integrated throughout the life cycle. Initiation of test planning occurs within the System Requirements activity and iterative refinement of the test plan occurs throughout subsequent life cycle activities. Initiation and refinement of test scenarios also occurs, as necessary. This progression results in alignment and traceability among requirements, the actual products, and the Test Plan. It also results in enabling successful execution of the Test Plan later in the life cycle.

This entire activity typically would not be eliminated from the SDLC Model, unless the required software item is acquired or purchased externally, but the tasks within the activity may be tailored. In some organizations, unit test tasks are planned informally and detailed unit test planning and scenarios are not required. However, even if unit test plans and scenarios are not formally developed and documented, an expectation exists that the unit will execute, as required, during the Integration Test. Quality problems and schedule delays in the Integration Test activity may indicate that tailoring of the unit test tasks be re-evaluated and that unit test tasks be formalized.

Performing tasks associated with this activity reduces project risk by reducing uncertainty in the implementation of the software and enhancing the ability to maintain the software. In deciding which tasks to include within this activity, project delivery and system development practitioners should consider the impact to the quality of the code to be delivered.

Avoidance and early resolution of flaws and problems positively impact the quality of and customer satisfaction with the implemented software. In addition, flaws and problems corrected early in software development have less of an impact on a project's schedule and budget.

Alignment with Enterprise Processes, shown in Figure 1, should be considered during Construction tasks, including coding and data modeling standards.

Entry Criteria

- Project Plan approved and baselined
- SyRS approved and baselined, if applicable
- SyDD approved and baselined, if applicable
- System Requirements allocated and documented in RTM, if applicable
- SRS approved and baselined
- SDD approved and baselined

Inputs

- SyRS baseline, if applicable
- SyDD baseline, if applicable
- RTM baseline
- SRS baseline
- SDD baseline
- Test Plan draft
- Test Scenario drafts
- Deployment Plan draft
- Existing system and software baselines
- Test Data
- Test Environment
- Organizational methodologies, standards, and best practices
- Existing SDLC deliverables

Activity Tasks

Task Number	Task	SDLC Tool
3.6.1	Analyze Software Design Description (SDD).	
3.6.2	Perform detailed design.	
3.6.3	Code software.	
3.6.4	Conduct code reviews.	• System Development Review Guidelines
3.6.5	Continue construction of Test Plan. Address the following information within the Test Plan: • Test Plan Overview • Test Methodology • Unit Test Phase	• Test Plan
3.6.6	Construct Unit Test Scenarios.	• Test Scenario
3.6.7	Prepare Unit Test environment.	
3.6.8	Prepare test data.	
3.6.9	Execute Unit Test.	
3.6.10	Validate test results.	
3.6.11	Construct Unit Test Report (See Reporting section of Test Plan).	• Test Plan
3.6.12	Update Test Plan, if applicable.	
3.6.13	Update Unit Test Scenarios, if applicable.	

Task Number	Task	SDLC Tool
3.6.14	Continue construction of Test Plan. Address the following information within the Test Plan, if applicable: • Test Plan Overview • Test Methodology • Unit Test Phase • Integration Test Phase • System Test Phase • Acceptance Test Phase • Test Schedule • Test Monitoring and Reporting	• Test Plan
3.6.15	Continue construction of Test Scenarios. Include traceability to requirements. Address the following scenarios: • Integration Test • System Test • Acceptance Test	• Test Scenarios • Requirements Traceability Matrix (RTM)
3.6.16	Capture baselines and/or place under Configuration Management: • Unit Tested Software baseline • Test Plan draft • Unit Test Scenarios baseline • Test Scenario drafts, other than Unit • RTM baseline • Unit Test Report • Test Data • Test Environment • Other deliverables, if applicable	

Exit Criteria

- Unit Test successful
- Units of software ready for Integration Test

Outputs

- Unit tested software baseline
- Test Plan draft
- Unit Test Scenarios baseline
- Test Scenario drafts, other than Unit
- RTM baseline
- Unit Test Report
- Test Data
- Test Environment
- Other deliverables, if applicable

3.7 INTEGRATION TEST

Purpose

To ensure that aggregates of units tested during the unit test phase can be successfully integrated into a software item, as designed.

Tailoring Considerations

Consider that tasks within this activity include preparation for and execution of the tasks documented in the Integration Test section of the Test Plan. This activity should be tailored to include tasks to integrate aggregates of units and determine if the aggregate satisfies requirements.

This activity typically would not be eliminated from the SDLC Model, unless the software item has only one unit or component that was tested during unit test. However, the tasks within this activity may be tailored for the particular project.

As stated in the Construction Activity Tailoring Considerations, if quality problems and schedule delays occur within the Integration Test activity, it may indicate that tailoring of the Unit Test tasks be re-evaluated and that unit test tasks be formalized.

Entry Criteria

- Software Unit Test successful
- Integration Test planning complete
- Integration Test Scenarios complete

Inputs

- Unit tested software baseline
- Test data
- Test environment
- SDD baseline
- RTM baseline
- Test Plan draft
- Integration Test Scenarios draft
- Previous test results, if applicable
- Other deliverables, if applicable
- Organizational methodologies, standards, and best practices

Activity Tasks

Task Number	Task	SDLC Tool
3.7.1	Conduct review of Test Plan and Integration Test Scenarios.	• System Development Review Guidelines • Test Planning Verification Checklist
3.7.2	Capture baselines and place under configuration management: • Test Plan • Integration Test Scenarios	

Task Number	Task	SDLC Tool
3.7.3	Prepare test environment.	
3.7.4	Prepare test data.	
3.7.5	Conduct review of test readiness.	• System Development Review Guidelines • Test Readiness Assessment Checklist
3.7.6	Execute Integration Test.	• Test Plan
3.7.7	Validate test results.	
3.7.8	Update Test Plan, if applicable.	• Test Plan
3.7.9	Update Integration Test Scenarios, if applicable.	• Test Scenario
3.7.10	Construct Integration Test Report (See Reporting section of Test Plan).	• Test Plan
3.7.11	Capture baselines and/or place under Configuration Management: • Integration tested software baseline • Test Plan baseline • Integration Test Scenarios baseline • RTM baseline • Integration Test Report • Test Data • Test Environment • Other deliverables, if applicable	

Exit Criteria

- Integration Test successful
- Software ready for System Test

Outputs

- Integration tested software baseline
- Test Plan baseline
- Integration Test Scenarios baseline
- RTM baseline
- Integration Test Report
- Test Data
- Test Environment
- Other deliverables, if applicable

3.8 SYSTEM TEST

Purpose

To ensure that the system performs according to documented requirements.

Tailoring Considerations

Consider that tasks within this activity include preparation for and execution of the tasks documented in the System Test section of the Test Plan. This activity should be tailored to include tasks to integrate and test all software items, hardware, manual processes, and other system interfaces that constitute the system, as designed.

This activity typically would not be eliminated from the SDLC Model, but the tasks within this activity may be tailored for the particular project.

Entry Criteria

- System component(s) ready for system test
- System Test planning complete
- System Test Scenarios complete

Inputs

- Integration tested software baseline
- Test data
- Test environment
- SyDD baseline
- RTM baseline
- Test Plan baseline
- System Test Scenarios draft
- Previous test results, if applicable
- Other deliverables, if applicable
- Organizational methodologies, standards, and best practices

Activity Tasks

Task Number	Task	SDLC Tool
3.8.1	Conduct review of Test Plan and System Test Scenarios, if applicable.	• System Development Review Guidelines • Test Planning Verification Checklist
3.8.2	Capture baselines and place under Configuration Management: • Test Plan • System Test Scenarios	
3.8.3	Prepare test environment.	
3.8.4	Prepare test data.	
3.8.5	Conduct review of test readiness.	• System Development Review Guidelines • Test Readiness Assessment Checklist

Task Number	Task	SDLC Tool
3.8.6	Execute System Test.	
3.8.7	Validate test results.	
3.8.8	Update Test Plan, if applicable.	• Test Plan
3.8.9	Update System Test Scenarios, if applicable.	• Test Scenario
3.8.10	Construct System Test Report (See Reporting section of Test Plan).	• Test Plan
3.8.11	Capture baselines and/or place under Configuration Management: • System tested software baseline • Test Plan baseline • System Test Scenarios baseline • RTM baseline • System Test Report • Test data • Test environment • Other deliverables, if applicable	

Exit Criteria

- System Test successful
- System ready for Acceptance Test

Outputs

- System tested software baseline
- Test Plan baseline
- System Test Scenarios baseline
- RTM baseline
- System Test Report
- Test data
- Test environment
- Other deliverables, if applicable

3.9 ACCEPTANCE TEST

Purpose

To ensure that the completed system performs according to the stakeholders' expectations based on specified requirements before the system becomes operational.

Tailoring Considerations

Consider that tasks within this activity include preparation for and execution of the tasks documented in the Acceptance Test section of the Test Plan. This activity should be tailored to include tasks to integrate and test all software items, hardware, manual processes, and other system interfaces that constitute the system, as designed, to determine if the system should be accepted.

This activity typically would not be eliminated from the SDLC Model, but the tasks within this activity may be tailored or combined with another stage of testing for the particular project.

Entry Criteria

- System Test successful
- System ready for Acceptance Test
- Acceptance Test planning complete
- Acceptance Test Scenarios complete

Inputs

- System tested software baseline
- Test data
- Test environment
- RTM baseline
- Test Plan baseline
- Acceptance Test Scenarios draft
- Previous test results, if applicable
- Other deliverables, if applicable
- Organizational methodologies, standards, and best practices

Activity Tasks

Task Number	Task	SDLC Tool
3.9.1	Conduct review of Test Plan and Test Scenarios, if applicable.	• System Development Review Guidelines • Test Planning Verification Checklist
3.9.2	Capture baselines and place under Configuration Management: • Test Plan • Test Scenarios	
3.9.3	Prepare Acceptance Test environment.	
3.9.4	Prepare Acceptance Test data.	

Task Number	Task	SDLC Tool
3.9.5	Conduct review of test readiness.	• System Development Review Guidelines • Test Readiness Assessment Checklist
3.9.6	Execute Acceptance Test.	
3.9.7	Validate test results.	
3.9.8	Update Test Plan, if applicable.	• Test Plan
3.9.9	Update Acceptance Test Scenarios, if applicable.	• Test Scenario
3.9.10	Construct Acceptance Test Report (See Reporting section of Test Plan).	• Test Plan
3.9.11	Capture baselines and/or place under configuration management: • Acceptance tested software baseline • Test Plan baseline • Acceptance Test Scenarios baseline • RTM baseline • Acceptance Test Report • Test data • Test environment • Other deliverables, if applicable	

Exit Criteria

- Acceptance Test successful
- System ready for deployment

Outputs

- Acceptance tested software baseline
- Test Plan baseline
- Acceptance Test Scenarios baseline
- RTM baseline
- Acceptance Test Report
- Test data
- Test environment
- Other deliverables, if applicable

3.10 DEPLOYMENT

Purpose

To complete deployment preparation and deployment of the system, as documented in the Deployment Plan.

Tailoring Considerations

Consider which tasks are required for assuring that the deployment plan is finalized, reviewed, approved, and executed, resulting in an operational system. Initiation of test and deployment planning occurs within the System Requirements activity and iterative refinement of those plans occurs throughout subsequent life cycle activities. This progression results in alignment and traceability among requirements, the actual products, and the plans. It also results in enabling successful execution of test and deployment plans later in the life cycle.

When deployment tasks are successful, the result is the implementation of the system designed based on the documented requirements. When catastrophic problems occur with the deployment, the contingency planning tasks documented within the Deployment Plan may be executed to mitigate or eliminate additional risk.

The Deployment activity typically would not be eliminated from the SDLC Model, but the tasks within this activity may be tailored for a particular project.

Entry Criteria

- Acceptance Test successful, with stakeholders accepting unresolved defects, if applicable
- All system components and documentation baselined and ready for deployment

Inputs

- System Components
- Deployment Plan draft
- Production Data, if applicable
- Production Environment
- System Documentation baseline
- Training Materials
- Other deliverables, if applicable
- Organizational methodologies, standards, and best practices

Activity Tasks

Task Number	Task	SDLC Tool
3.10.1	Finalize Deployment Plan.	• Deployment Plan
3.10.2	Conduct Deployment Readiness Review.	• System Development Review Guidelines • Deployment Readiness Assessment Checklist
3.10.3	Capture Deployment Plan baseline and place under configuration management.	

Task Number	Task	SDLC Tool
3.10.4	Obtain Approval/Acceptance to Deploy.	
3.10.5	Initiate Deployment Plan Execution. Perform activities such as: • Prepare Production Environment • Prepare Software Release(s) • Execute Physical Configuration Audit • Create/Convert Production Data • Prepare, Document, and Conduct Training • Prepare Notification of Deployment	• System Development Review Guidelines • Physical Configuration Audit Checklist
3.10.6	Capture baselines and place under configuration management: • Software Release(s) • Physical Configuration Audit Report • Other deliverables, if applicable	
3.10.7	Complete Deployment Plan Execution. Perform activities such as: • Prepare Production Environment • Prepare Software Release(s) • Execute Physical Configuration Audit • Create/Convert Production Data • Prepare, Document, and Conduct Training • Deploy System Components • Execute Contingency Plan, if necessary • Distribute Notification of Deployment	
3.10.8	Capture baselines and/or place under configuration management: • Deployment Plan baseline • System Components/Software baselines • Data Creation/Conversion Software baseline • Physical Configuration Audit Report • Training Materials baseline • Notification of Deployment • Other deliverables, if applicable	

Exit Criteria

- Deployment Plan baseline
- Approval/Acceptance to Deploy
- Successful system deployment or successful execution of contingency plan tasks
- System components/software releases and documentation baselined
- Data created/converted
- Data creation/conversion software baselined
- Training conducted and materials baselined

Outputs

- Deployment Plan baseline
- Approval/Acceptance to Deploy
- Production Environment
- System Components/Software baselines
- Data Creation/Conversion Software baseline

- Physical Configuration Audit Report
- Production Data
- Training Materials baseline
- Notification of Deployment
- Other deliverables, if applicable

Appendix A: Examples of System Development Life Cycle Models

A.1 INCREMENTAL SYSTEM DEVELOPMENT LIFE CYCLE MODEL

An incremental system development life cycle model is a model that uses scheduling and staging strategies that allow pieces of the system to be developed at different times or rates and integrated as they are completed. Between increments, additions may be made to the requirements, process changes could be incorporated, or the schedule could be improved and revised. Incremental is distinguished from iterative development in that the latter supports predicted rework of parts of the system.

Incremental development is a technique that recognizes and takes advantage of the fact that requirements analysis precedes design, design precedes coding, coding precedes testing, and code testing precedes final validation against the initial requirements. Using the incremental system development life cycle model, development is broken into smaller development efforts and end products are integrated and/or released as they are completed. End products and duration of each increment are monitored, building an increasingly accurate picture of the size of the development effort and the rate of progress. Each increment follows its own validation "V", as illustrated in the V-model, and the development effort as a whole becomes the management of the series of *V*s. As depicted in Figure 5, *V*s fit within *V*s for increments that aid in managing visibility and risk. The development process is examined and possibly revised after selected increments.

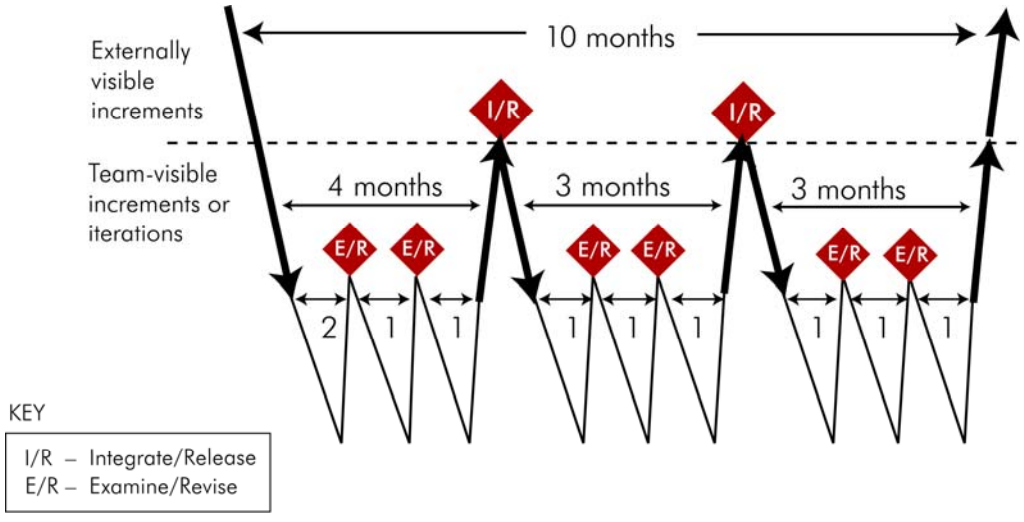


Figure 5. Illustration of an Incremental System Development Life Cycle Model

A.2 RAPID APPLICATION DEVELOPMENT SYSTEM DEVELOPMENT LIFE CYCLE MODEL

Rapid Application Development (RAD) is a system development life cycle model that focuses on building systems in a short amount of time; traditionally with compromises in usability, features, and/or execution speed. The RAD model compresses traditional system development life cycle models to develop high-quality products quickly by:

- Gathering requirements
- Prototyping and early, reiterative testing of designs
- Re-using software components
- Utilizing powerful development software such as CASE tools, prototyping tools, and code generators
- Deferring design improvements to subsequent product versions in order to meet rigidly paced schedule
- De-emphasizing formality in reviews and other team communication

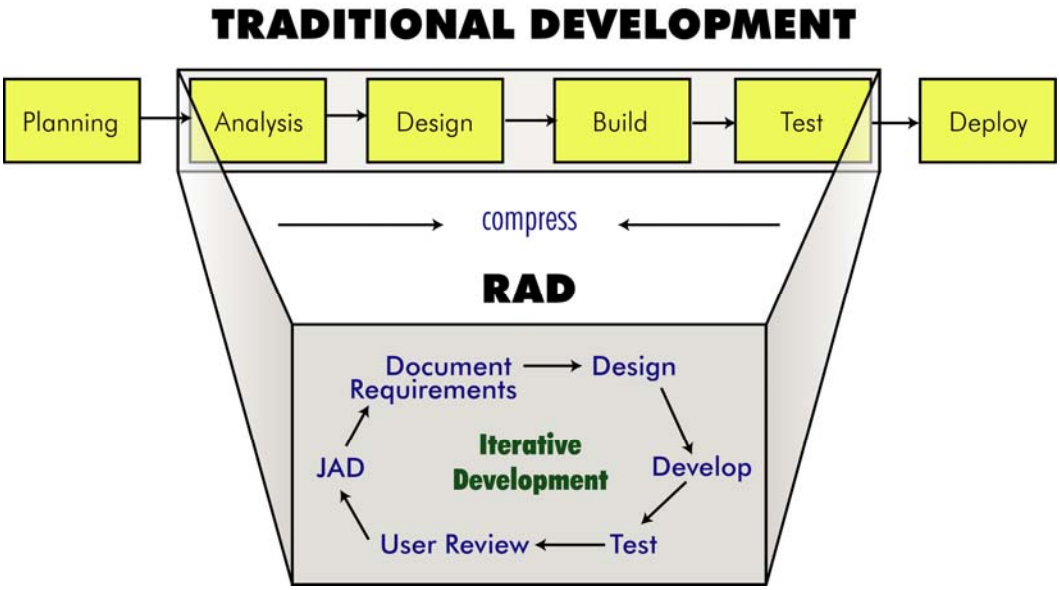


Figure 6. Illustration of a Rapid Application Development System Development Life Cycle Model

A.3 AGILE SYSTEM DEVELOPMENT LIFE CYCLE MODEL

The Agile system development life cycle model embraces and promotes evolutionary change throughout the entire development life cycle and attempts to minimize risk by performing development in short intervals, called iterations. Agile methods emphasize real-time communication, preferably face-to-face, over written documents. Agile methods also emphasize working software as the primary deliverable and measure of progress.

The Agile system development life cycle model begins with initial elicitation of requirements in order to conceptualize the system. Once the system concept is formed, requirements are estimated and prioritized in order to plan the development iterations. Within each iteration, requirements continue to evolve and are used in brief model storming sessions in order to design the functionality required to meet requirements allocated to the iteration. When the system model is developed/modified, the coding is initiated/continued using a test-driven approach. Once the functionality for the iteration is tested successfully, refactoring takes place. Refactoring is a disciplined technique for altering the structure of an existing code base without altering functionality. When refactoring is completed and functionality is confirmed, the code is deployed internally, completing the iteration.

If plans include deploying the code beyond the development iteration, the code is then deployed externally. Eventually, all iterations are completed, resulting in a system with full functionality that is accepted by stakeholders because it meets allocated requirements.

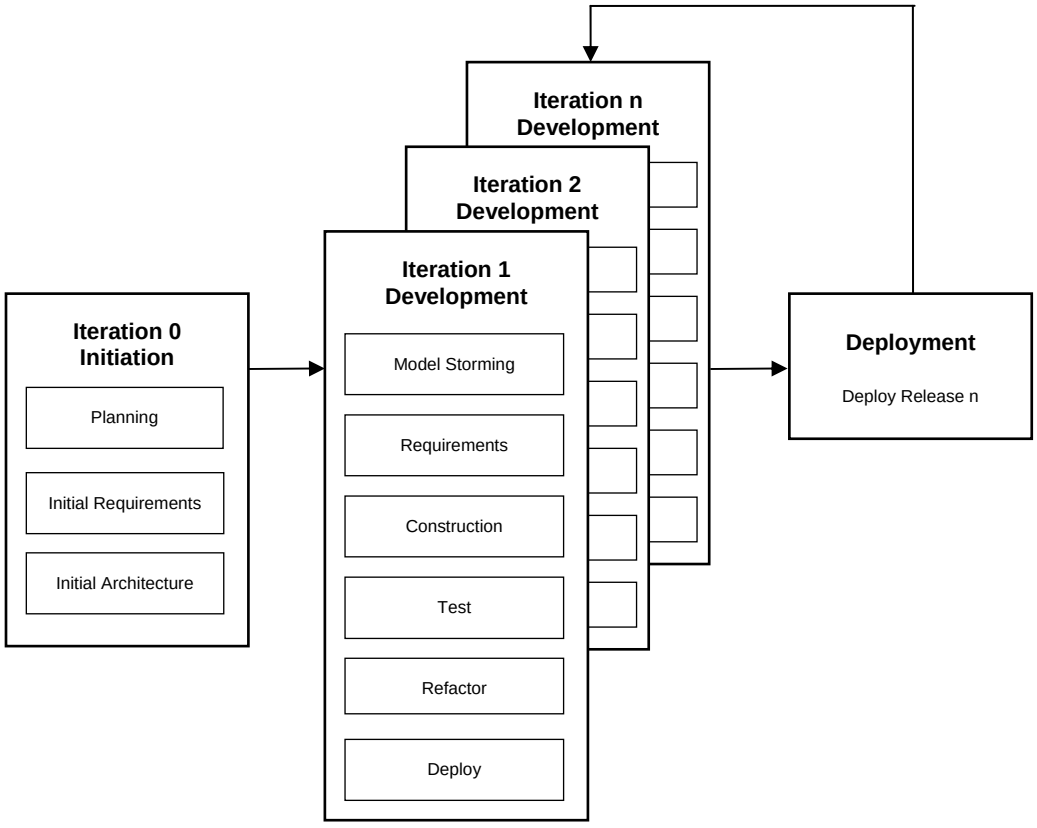
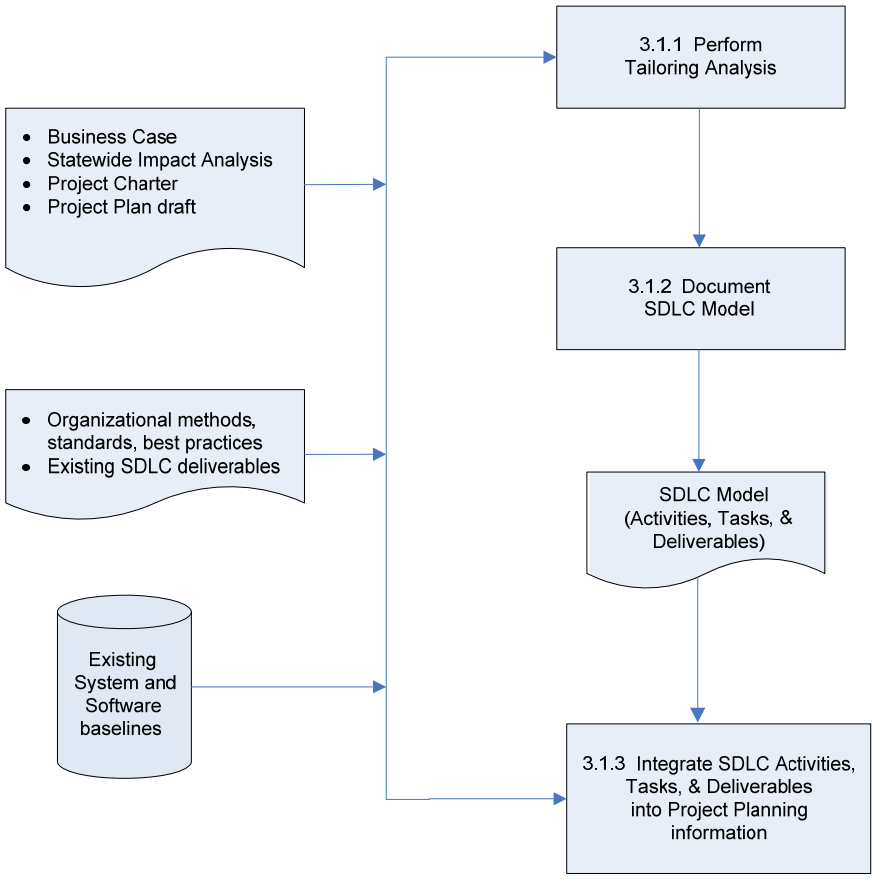
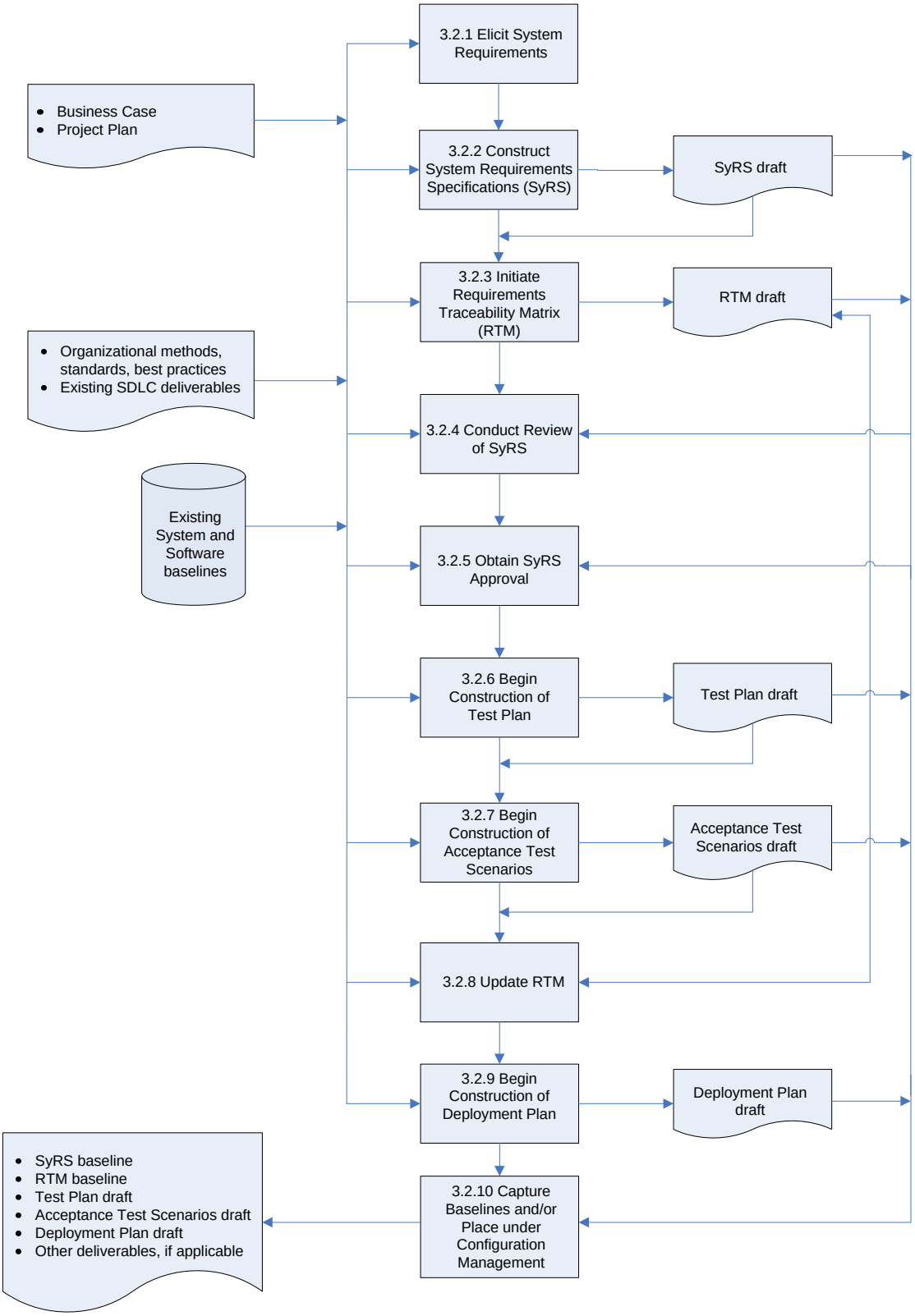


Figure 7. Illustration of Agile System Development Life Cycle Model

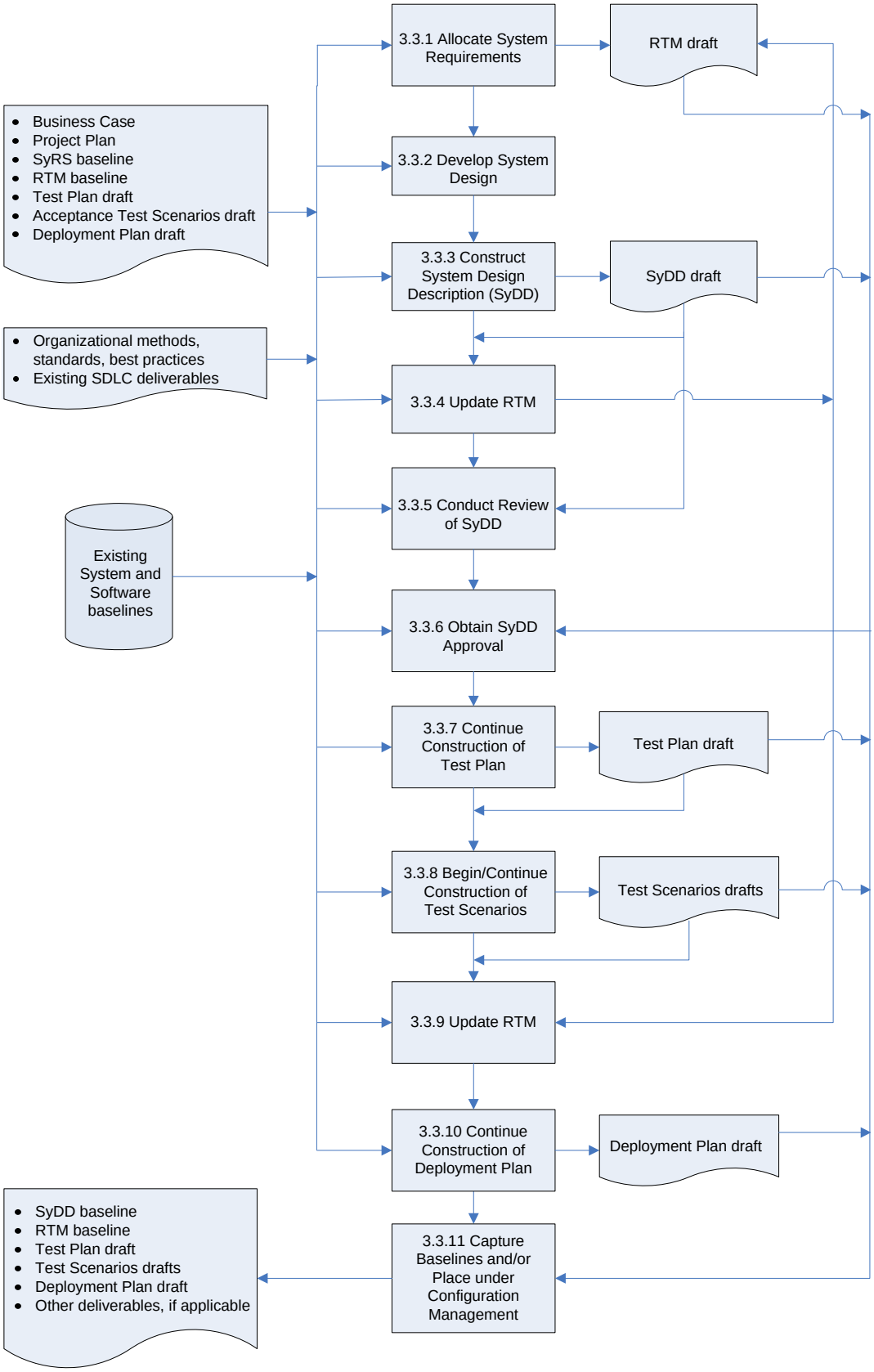
Appendix B: Activity 3.1 – Development Process Tailoring Flow Chart



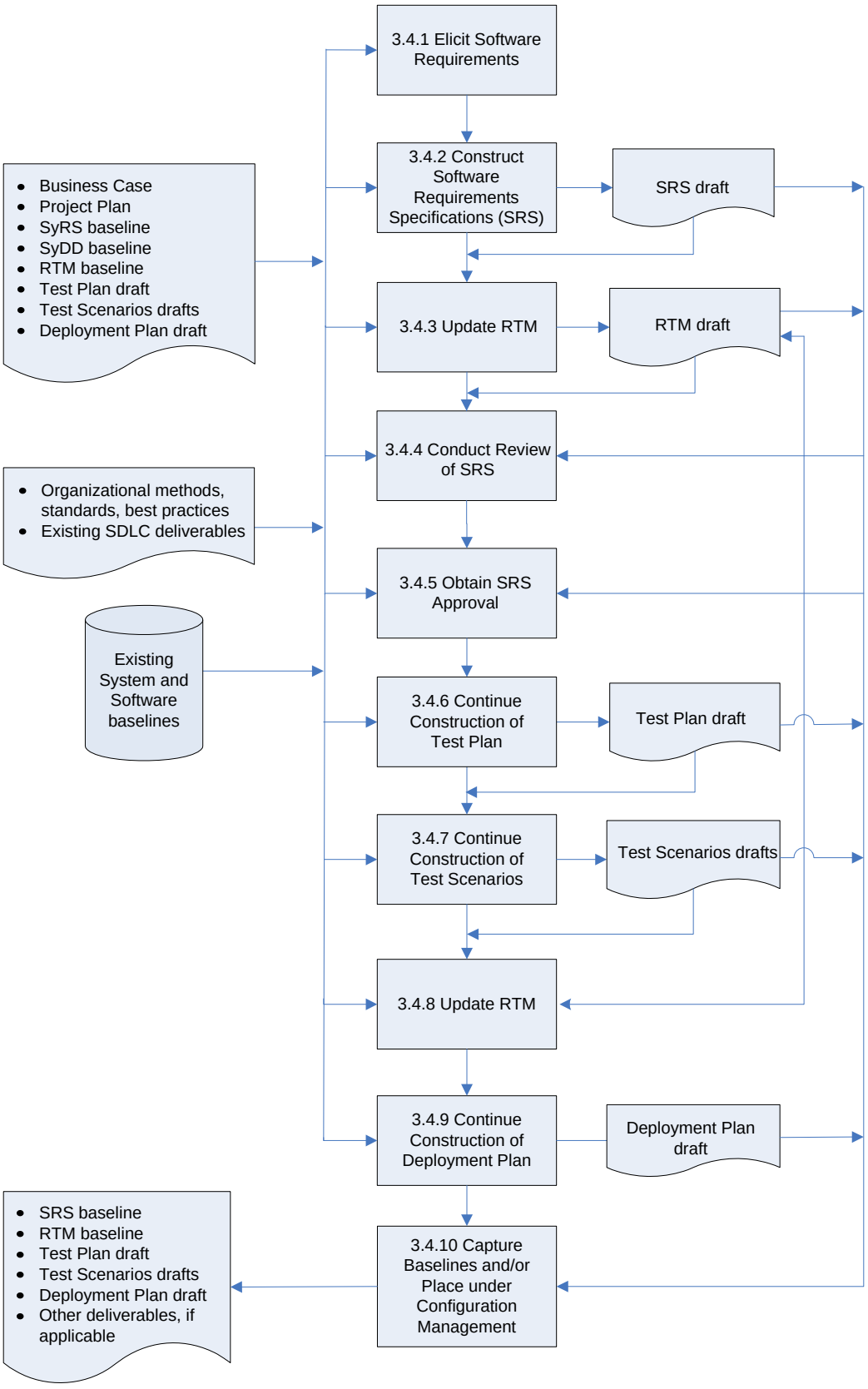
Appendix C: Activity 3.2 – System Requirements Flow Chart



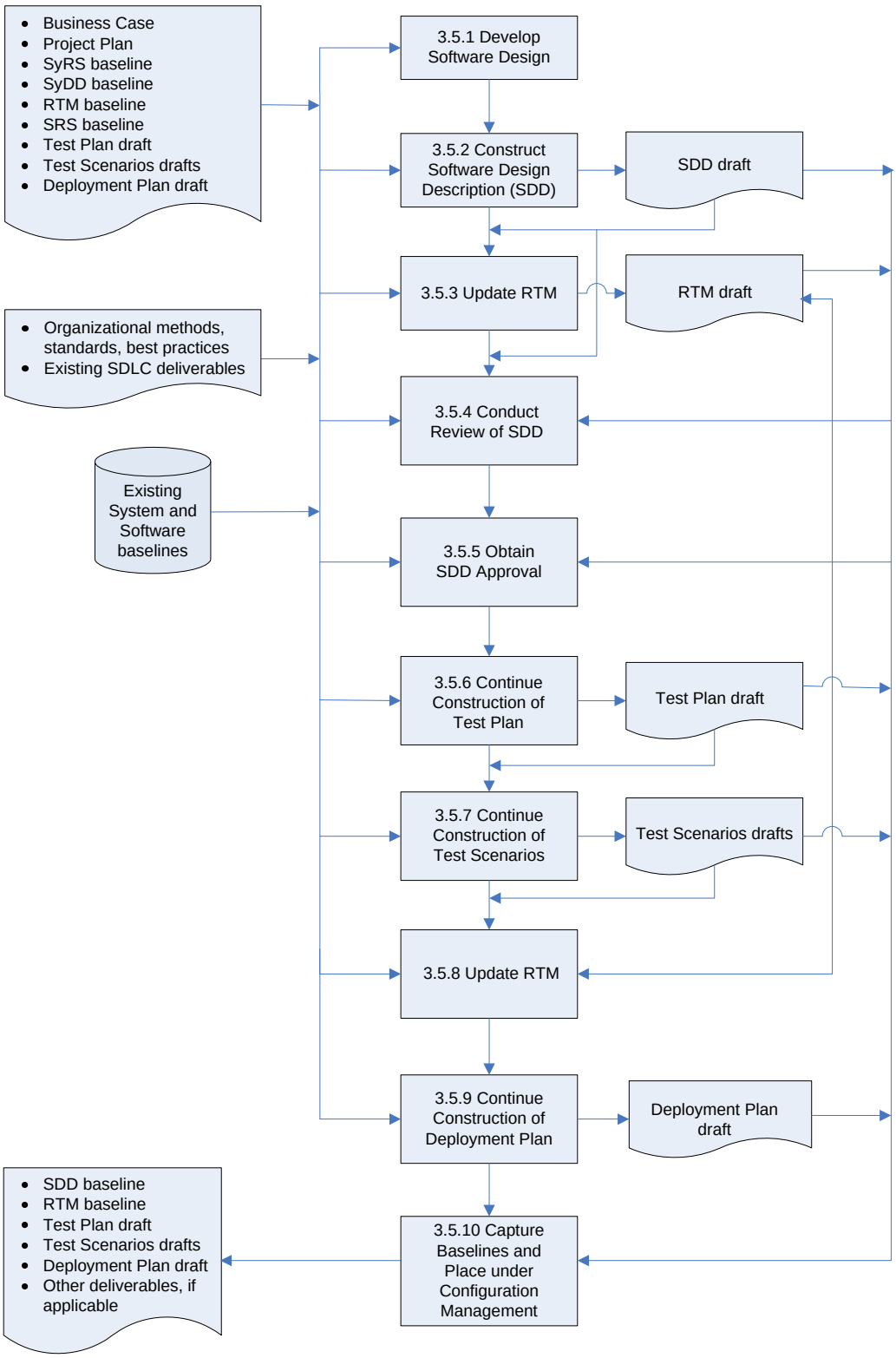
Appendix D: Activity 3.3 – System Design Flow Chart



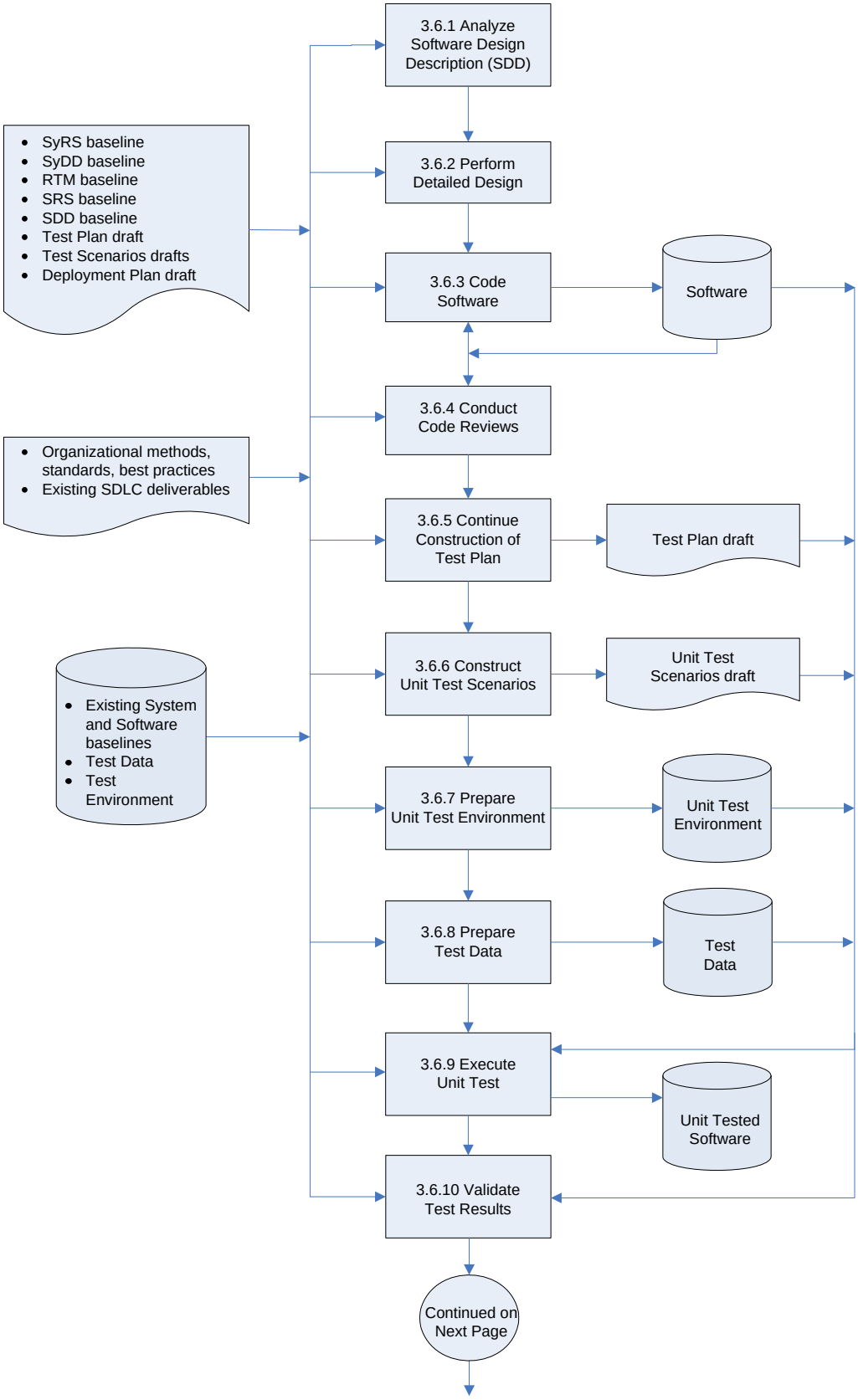
Appendix E: Activity 3.4 – Software Requirements Flow Chart



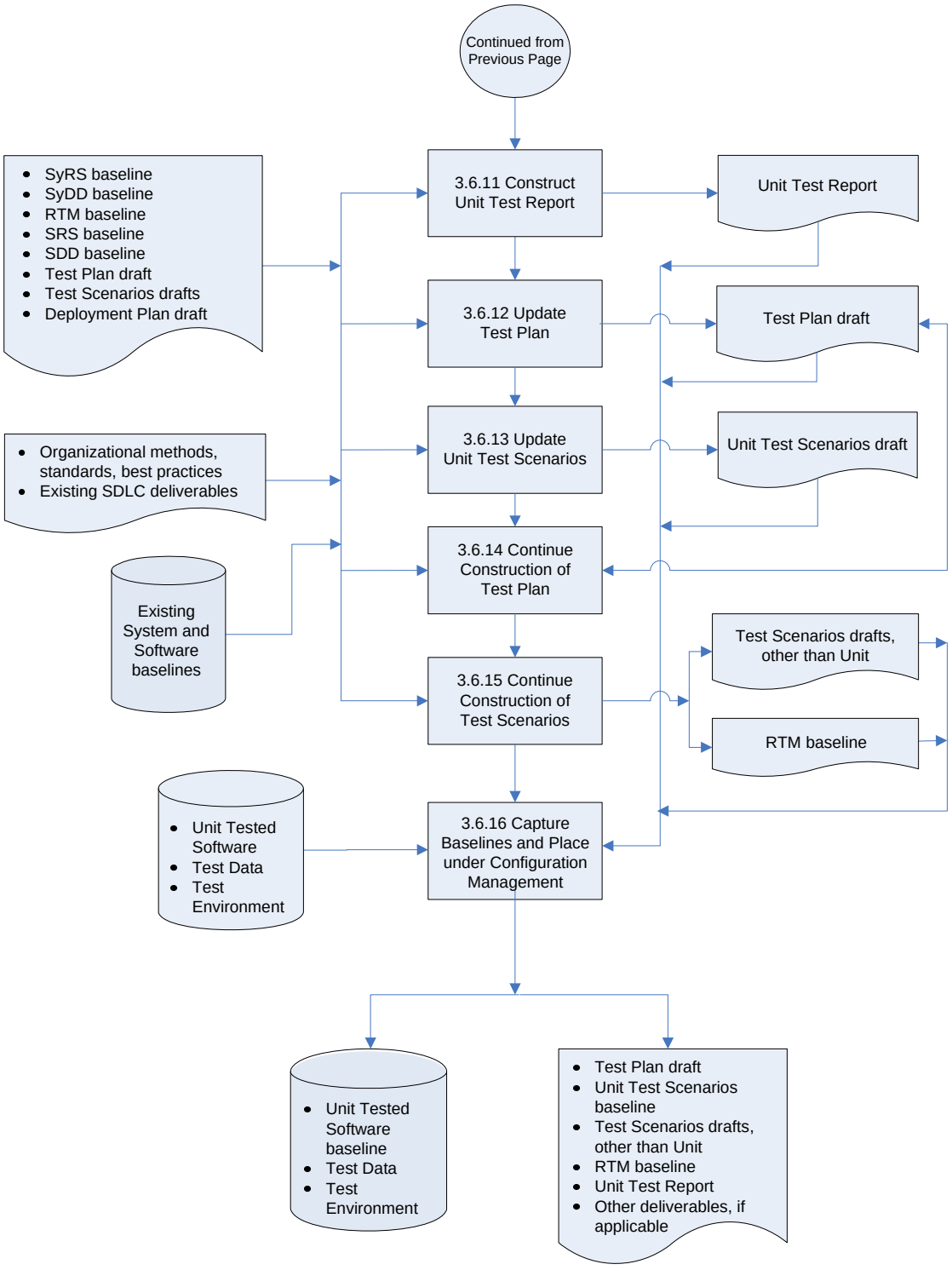
Appendix F: Activity 3.5 – Software Design Flow Chart



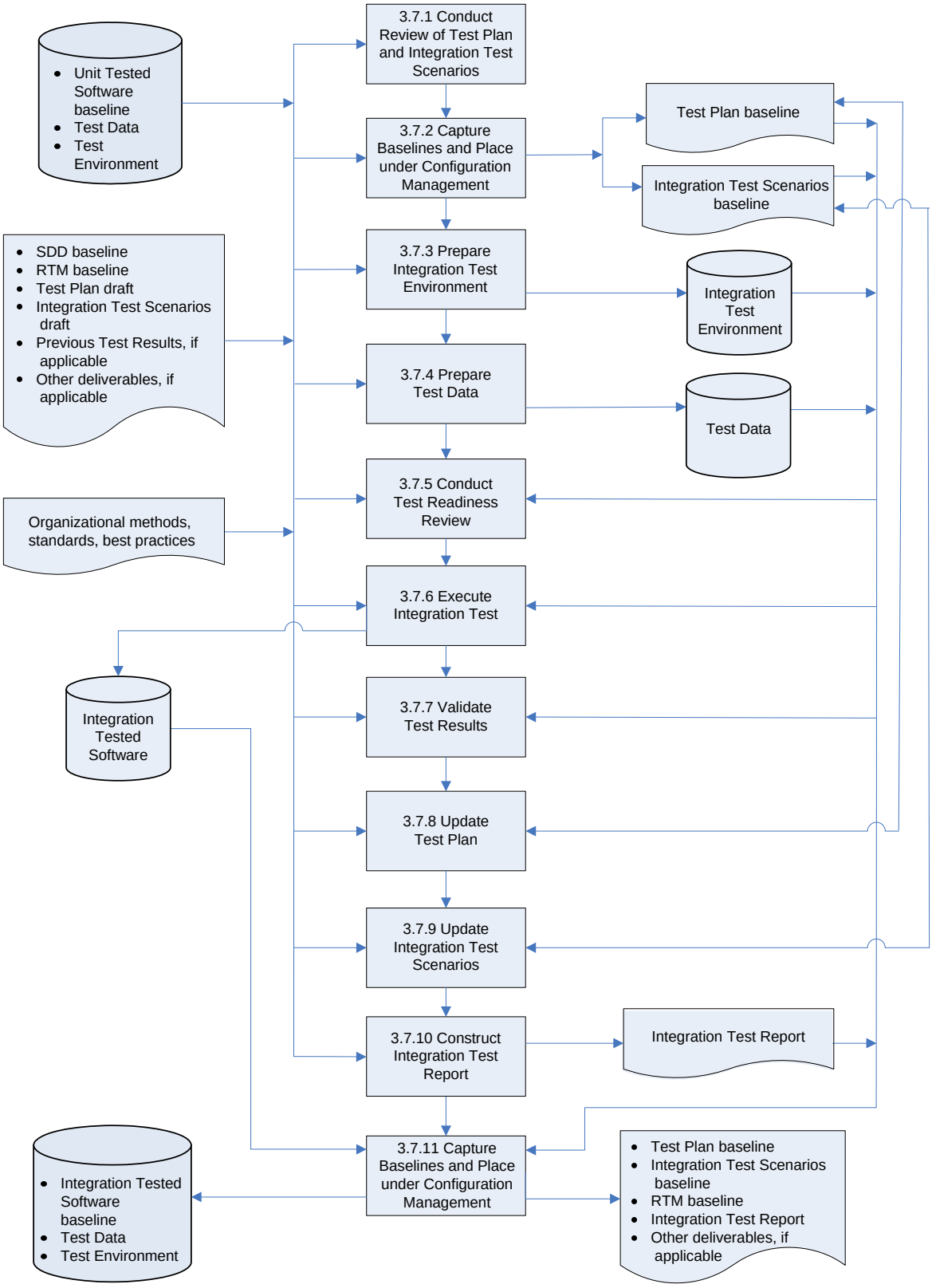
Appendix G: Activity 3.6 – Construction Flow Chart



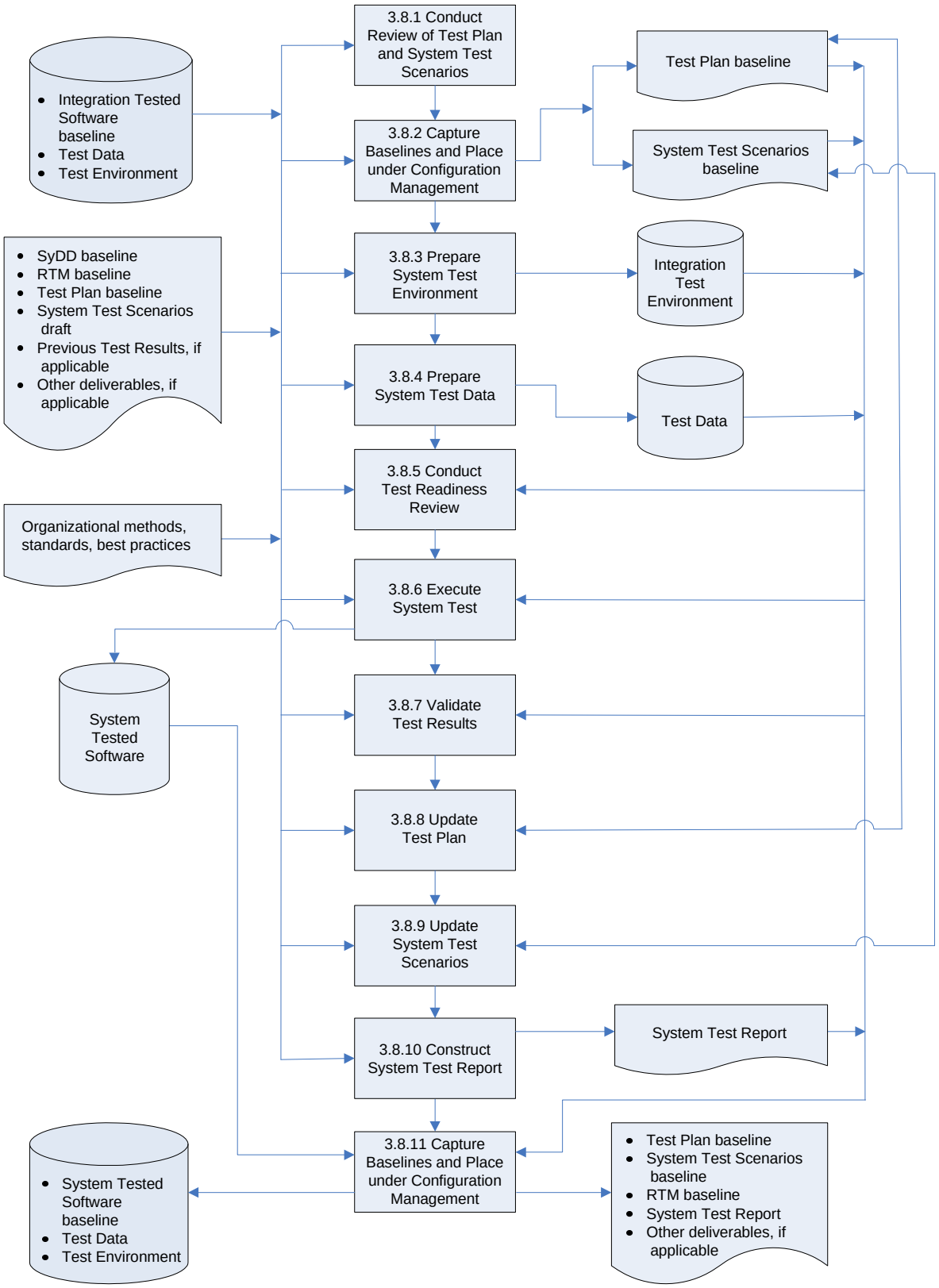
Appendix G: Activity 3.6 – Construction Flow Chart, continued



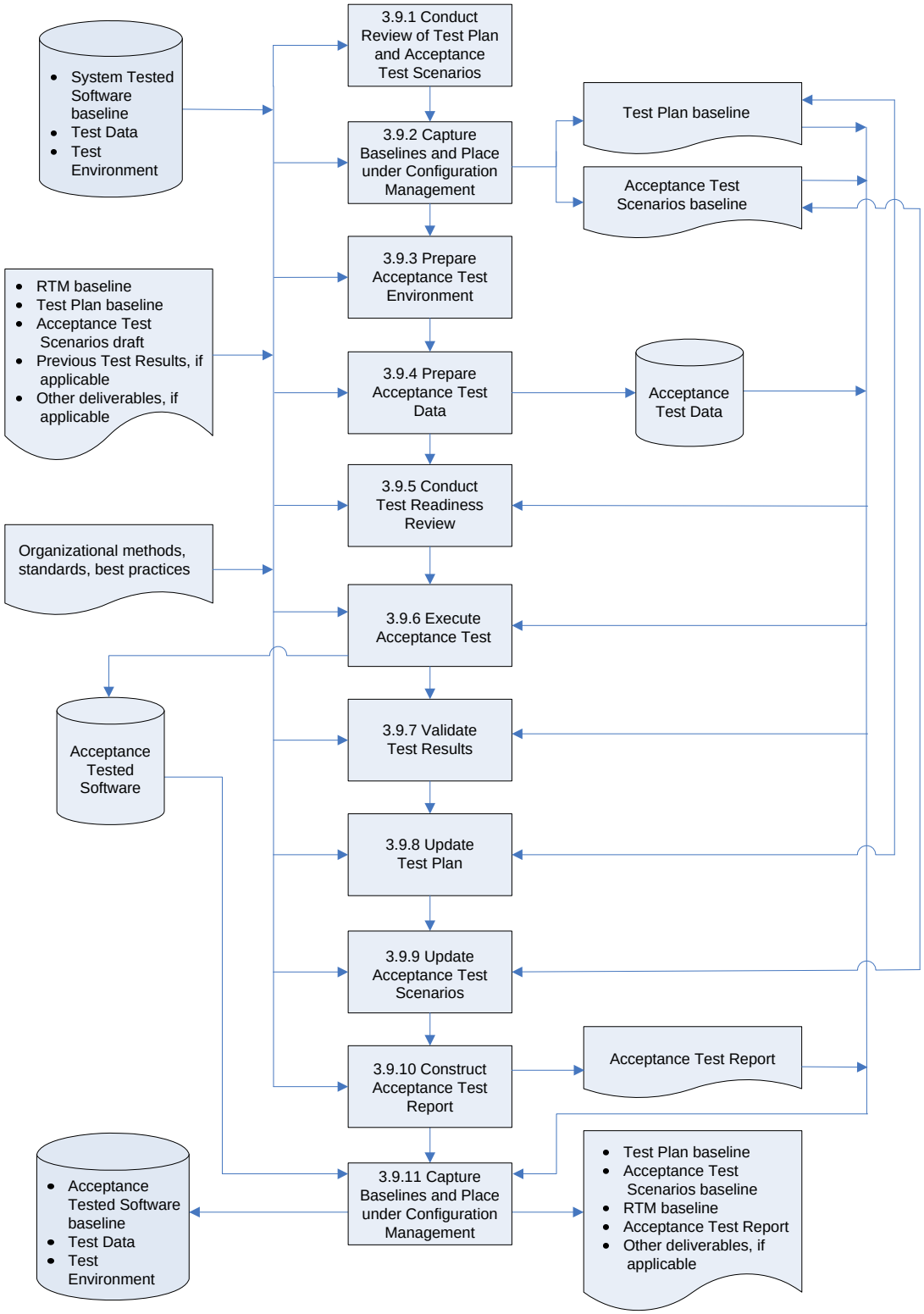
Appendix H: Activity 3.7 – Integration Test Flow Chart



Appendix I: Activity 3.8 – System Test Flow Chart



Appendix J: Activity 3.9 – Acceptance Test Flow Chart



Appendix K: Activity 3.10 – Deployment Flow Chart