COMMONWEALTH OF PENNSYLVANIA DEPARTMENT OF HUMAN SERVICES

INFORMATION TECHNOLOGY GUIDELINE

Name Of Guideline: **System Development Methodology (SDM)**	Number: **GDL-EPPM013**
Domain: **Business**	Category: **System Development Methodology**
Date Issued: **03/01/1999**	Issued By: **DHS Bureau of Information Systems**
Date Revised: **03/29/2016**	

Table of Contents

System Development Methodology (SDM)

1.0 Introduction

Systems Development Methodology (SDM)

The Systems Development Methodology (SDM) embodies all facets of bringing business ideas to reality or resolving business problems that require information systems and technology (IS/IT) as a solution enabler. The SDM is the consortium of complementary and interdependent governance frameworks, methodologies, and models that are used to direct, manage, and execute the development and delivery of business information systems solutions. The guiding principles of the SDM are:

- Ensure alignment of Business and IT
- Consider the value proposition relative to the business and citizens
- Meet or exceed customer expectations
- Deliver cost effective, timely and quality technology solutions
- Establish effective and efficient processes and procedures with performance metrics to evaluate performance and facilitate continuous improvement.
- Effective utilization of IS/IT assets as well as establishing relevant standards, procedures, methods, models, and industry best practices to design, develop, and deploy sound business solutions and optimize support for our customers.

One methodology does not fit all sizes and types of system development efforts. Therefore, the Department of Human Services (DHS) SDM methodology provides options for project teams to use various software and systems development process models ranging from traditional water fall sequential to more flexible iterative variations or alternative approaches depending on the type, scope, complexity, risk tolerance, and cost of the project initiative. It also provides a modified SDLC to accommodate the acquisition and implementation of commercial-off-the-shelf (COTS) products and transfer technology solutions.

Purpose, Scope, and Applicability

The SDM serves as the mechanism to ensure the systems under development meet the established requirements and support DHS's mission functions. It provides a structured approach to managing IT projects beginning with establishing the justification for initiating a systems development or maintenance effort and concluding with system disposition.

The primary audiences for this methodology are the application developers, IT project managers, and project sponsors/users responsible for defining and delivering the DHS systems, their staff, and their support contractors. Specific roles and responsibilities are described throughout each phase.

SDM Objectives

This guide was developed to disseminate proven practices to application developers, project managers, and project sponsors/users throughout the DHS. The specific objectives expected include:

- To consider the business drivers and associated value proposition
- To reduce the risk of project failure
- To consider various systems and technology engineering process parameters/options for the business solution
- To consider system and data requirements throughout the entire life of the system
- To identify technical and management issues early
- To disclose all life cycle costs to guide business decisions
- To foster realistic expectations of what the systems will and will not provide
- To provide information to better balance programmatic, technical, management, and cost aspects of proposed system development or modification
- To encourage periodic evaluations to identify systems that are no longer effective
- To measure progress and status for effective corrective action
- To support effective resource management and budget planning
- To consider meeting current and future business requirements

SDM Framework

The SDM framework is outlined below. These are the foundations for developing business solutions.

- **Software and Technology Engineering Process**

  The systems and technology engineering process (STEP) is used to ensure a structured approach to information systems design, development, delivery, maintenance, and operation. The STEP outlines a structured framework for bringing business concepts that require information systems and technology solutions to fruition. The STEP methodologies, processes, and procedures (i.e., SDLC, CMM, SDPM, and IEEE,) combined with project development life cycle and other components (i.e., ARB, ITIL, EA, SQA, etc) define and govern the DHS Systems Development Methodology (SDM). The SDM describes an overall structured approach to information management. Primary emphasis is placed on the information and systems decisions to be made and the proper timing of decisions. It provides a flexible framework for approaching a variety of systems projects. The framework enables application developers, project managers, and project sponsors/users to combine activities, processes, and products, as appropriate, and to select the tools best suited to the unique needs of each project. Regardless of the Software Development Process Model (SDPM) used, they all have common SDLC elements: Feasibility, Requirements, Design, Develop, Test, and Implementation. Hence, this document looks at each of these elements in more detail and outlines the key attributes that must be considerations for all systems solutions.

- **Project Management:**

  Business solution initiatives will require the utilization of sound project management methodologies. The business product or solution from idea to fruition will be managed and executed through the project life cycle phases (i.e., Initiation, planning, Execution, Control/Monitoring, Closeout with risk/issues and change management) and be required to incorporate the following:

  i. **Each System Project Must Undergo Formal Acceptance**

     The project sponsor identifies the representative who will be responsible for formally accepting the delivered system at the end of the Acceptance and Installation Phase.

  ii. **Secure a Project Sponsor**

     To help ensure effective planning, management, and commitment to information systems, each project must have a clearly identified program sponsor. The project sponsor serves in a leadership role, providing guidance to the project team and securing, from senior management, the required reviews and approvals at specific points in the life cycle. Senior management approval authority may be varied based on dollar value, visibility level, congressional interests or a combination of these. The project sponsor is responsible for identifying who will be responsible for formally accepting the delivered system at the end of the Acceptance and Installation Phase.

  iii. **Establish Project Management Team**

     The establishment of a Project Management Team can aid in the success of a project. A Project Management Team is a multidisciplinary group of people who support the Project Manager in the planning, execution, delivery and implementation of life cycle decisions for the project. The Project Management Team is composed of qualified empowered individuals from all appropriate functional disciplines that have a stake in the success of the project. Working together in a proactive, open communication, team oriented environment can aid in building a successful project and providing decision makers with the necessary information to make the right decisions at the right time.

  iv. **Project Manager must be selected for Each System Project**

     The Project Manager has responsibility for the success of the project and works through a project team and other supporting organization structures, such as working groups or user groups, to accomplish the objectives of the project. Regardless of organizational affiliation, the Project Manager is accountable and responsible for ensuring the project activities and decisions consider the needs of all organizations affected by the system.

  v. **A Comprehensive Project Plan is required for Each System Project**

     The project plan is a pivotal element in the successful solution of an information management requirement. The project plan must describe how each phase will be accomplished to suit the specific characteristics of the project. The project plan is a vehicle for documenting the project scope, tasks, schedule, allocated resources, and interrelationships with other projects. The plan is used to provide direction to the many activities of the life cycle and must be refined and expanded throughout the life cycle.

  vi. **Obtaining the Participation of Skilled Individuals is Vital to the Success of the System Project**

The skills of the individuals participating in a system project are the single most significant factor for ensuring the success of the project. The SDM is not intended as a substitute for information management skills or experience. While many of the skills required for a system project are discussed in later sections, the required skill combination will vary according to the project. All individuals participating in a system development project are encouraged to obtain assistance from experienced information management professionals.

vii. **Specific Individuals must be Assigned to Perform Key Roles throughout the Project Life Cycle and specific phases of the approved SDPM.**

Certain roles are considered vital to a successful system project and at least one individual must be designated as responsible for each key role. Assignments may be made on a full or part-time basis as appropriate. Key roles include program/functional management, quality assurance, security, telecommunications management, data administration, database administration, logistics, financial, systems engineering, test and evaluation, contracts management, and configuration management. For most projects, more than one individual should represent the actual or potential users of the system (that is, program staff) and be designated by the Project Manager of the program and organization.

viii. **Documentation of Activity Results and Decisions for Each Phase of the SDPM Life Cycle are Essential**

Effective communication and coordination of activities throughout the life cycle depend on the complete and accurate documentation of decisions and the events leading up to them. Undocumented or poorly documented events and decisions can cause significant confusion or wasted efforts and can intensify the effect of turnover of project management staff. Activities are not to be considered complete, nor decisions made, until there is tangible documentation of the activity or decision. For some large projects, advancement to the next phase cannot commence until the required reviews are completed and approved by senior management.

ix. **A System Project may not Proceed until Resource Availability is Assured**

Beginning with the approval of the project, the continuation of a system is contingent on a clear commitment from the project sponsor. This commitment is embodied in the assurance that the necessary resources will be available, not only for the next activity, but as required for the remainder of the life cycle.

- **Enterprise Architecture (EA)**

The DHS EA provides a common conceptual framework and Information Resource Management (IRM) standards all DHS organizations use to coordinate the acquisition, development, and support of information systems. These standards and architectural objectives advance DHS's ability to implement systems that are more interoperable and maintainable. The applicable EA frameworks associated with DHS SDM are: a) Business Reference Model (BRM: defines the business functions, services, and processes), b) Application Reference Model (ARM: defines all the software applications used to support the business), c) Data Reference Model (defines and classifies all the enterprise data and information assets associated with BRM and ARM), d) Technology Reference Model (TRM: defines all the technology layers, domains, service oriented architecture (SOA) and other affiliated strategic blueprints and roadmaps), d) Governance Reference Model (GRM: Defines the strategic, tactical, and operational governance frameworks to effectively direct and manage all EA models). The DHS EA blueprint is a comprehensive document that incorporates all the EA reference models with integrated enterprise security components.

- **Information Technology Infrastructure Library (ITIL)**
  ITIL is a set of concepts and practices for Information Technology Services Management (ITSM), Information Technology (IT) development and IT operations. The ITIL concepts relative to the DHS SDM are: a) Demand and Service Portfolio Management, b) Release and Deployment Management, c) Service Asset and Configuration Management.

- **Architecture Review Board (ARB)**

  The ARB is an integral component of the SDM that acts as phase gates throughout the systems development life cycle. There are four Architecture Review Boards that are utilized: a) ARB1 (to evaluate and clarify the business requirements; alignment with BRD), b) ARB2 (to evaluate and clarify the requirements translations, general design approach and conceptual models; alignment with GSD and SRD), ARB3 (to evaluate detailed systems design components, challenges and/or variations to original design approach, and new technology integrations; alignment with DSD), ARB4 (to evaluate operational readiness from a business and technology perspective; alignment with operational readiness certifications, checklist, Go/No Go approvals, and deployment playbook)

- **Quality Control**

  Quality Control (QC) refers to quality related activities associated with the creation of project deliverables. Quality control is used to verify that deliverables are of acceptable quality and that they are complete and correct. Examples of quality control activities include inspection, deliverable peer reviews and the testing process. Quality control is about compliance to SDM and other applicable policies and standards. Quality assurance is generic and does not concern the specific requirements of the product being developed.

1.1 Methodology Utilization

Description

The DHS methodology integrates software development, project management, and quality control and assurance practices into a flexible, yet orderly approach. It can be adapted to accommodate the specific needs of any DHS project. All computing platforms used in the department, including web, mainframe and client/server machines, may be incorporated.

The methodology presented here does not supersede, replace, or override more stringent requirements applied to specific projects, such as those receiving federal funding.

Questions

Questions concerning the interpretation or application of the SDM are submitted to the BIS Division of Enterprise Program and Portfolio Management (DEPPM) for clarification. Deviations and/or changes to the SDM require BIS approval. The project sponsor, users, and other project stakeholders must concur with any adaptations that may impact ongoing project initiatives.

DEPPM will analyze change requests, questions, and issues, and consult with the appropriate BIS stakeholders and then issue a response in one of the following ways:

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

1.2 Submitting Change Requests

DHS's information systems environment is continuously changing as emerging technologies are integrated into projects, user requirements are expanded, and organizational needs evolve. The DHS SDM will be expanded and revised, as needed, to reflect changes in the environment; also, improvements suggested through user feedback and the maturation of software engineering capabilities will prompt future modifications.

Users of the methodology are encouraged to submit suggestions for improving its content and to report any practices difficult to understand or which cause an implementation problem.

The key components of the SDM are outlined in the following subsections. These are the essential steps for developing sound business solutions.

- 2.0 Software and Technology Engineering process
- 3.0 Feasibility and Planning
- 4.0 Requirements Definition
- 5.0 General Systems Design
- 6.0 Detailed Systems Design
- 7.0 Development
- 8.0 Software Integration Testing
- 9.0 Acceptance and Installation
- 10.0 Operational Support

2.0 Software and Technology Engineering Process (STEP)

Purpose

The DHS Software and Technology Engineering Process (STEP), encompasses various models, standards, and technologies associated with the engineering of information systems and technology solution development and delivery process. The long-range goal has been to find repeatable, predictable processes that improve productivity and end product quality. The DHS STEP combined with project development life cycle define and govern the DHS Systems Development Methodology (SDM).

Scope

The STEP and approved software development process models are to be used for all the DHS information systems and applications. It is applicable across all information technology (IT) environments (e.g., mainframe, client, and server) and applies to contractually developed as well as in-house developed applications. The specific participants in the SDM process, and the necessary reviews and approvals, vary from project to project. The guidance provided, should be tailored to the individual project based on cost, complexity, and criticality to the agency's mission.

Applicability

The STEP is composition of industry standards, methodologies, and best practices that is integrated into the DHS SDM which can be applied to all information systems and technology solution initiatives to support business programs and operations. All project managers and development teams involved in system development projects represent the primary audience for the DHS SDM.

Introduction to the Software and Technology Engineering Process

The following paragraphs outline the DHS (BIS) STEP components: It outlines a structured framework for bringing business concepts that require information systems and technology solutions to fruition. The STEP methodologies, processes, and procedures (i.e., SDLC, CMM, SDPM, and IEEE) combined with project development life cycle define and govern the DHS Systems Development Methodology (SDM). The following paragraphs are a brief description of the core STEP components:

I. **Systems Development Life Cycle (SDLC)**: The SDLC is a universal model and integral component in the software and technology engineering process (STEP). In software and technology engineering, the SDLC model underpins many kinds of software development process models (SDPM) (i.e., Traditional, RAD, AGILE, etc). The main idea of the SDLC is to provide the underlining frameworks for the development of information systems and technology platforms that start from inception of the idea to final delivery of the business solution system. This and other associated software and technology engineering concepts combined with the project management life cycle methodologies form the DHS SDM framework for planning and controlling the creation of information systems and specifically outlines the framework for the information systems solution development (i.e., software/hardware)and delivery processes with integrated software quality assurance components. The descriptions of Systems Development Life Cycle (SDLC) phases are as follows:

1) **<u>Feasibility Analysis Phase</u>:** This phase focuses on aligning business and technical entities relative to: strategies, goals and objectives, planning, recourses, competing priorities, constraints, and overall value proposition involving the creation of new and/or modifications to existing DHS core enterprise Information Systems platforms required to support agency mission and specific program office business operations. Based on scope, complexity, and funding parameters, projects initiatives must first be evaluated, prioritized, and authorized by the Program Offices and BIS Portfolio Managers via the Business Review Board (BRB) process and aligned with the Communities of Practice (COP) goals and objectives. This phase is synchronized with the Business Review Board (BRB) and project initiation with High Level Estimating (HLE), high level scoping and planning activities. Once the project is prioritized and approved, the business and technical teams conduct more detailed assessments and planning activities mapping out the detailed project plan once the scope and the requirements have been further refined.

2) **<u>Requirement Analysis & Definition Phase</u>:** All possible requirements of the system to be developed are captured in this phase. Requirements are a set of functional and nonfunctional systems specifications and constraints that are expected from the system. The business and technical requirements are gathered through consultation sessions, these requirements are analyzed relative to category, criticality, prioritization, and verified for their validity and the possibility of incorporating the requirements in the system to be development. This phase of the SDLC produces a Business Requirement Specification Document (BRD) that outlines the business requirements. The BRD serves the purpose of guideline for the General Systems Design phase of the model.

3) **<u>System & Software Design Phase</u>:** The design phase is partitioned into two separate but interdependent components: 1) General Systems Design (GSD) and 2) Detailed Systems Design (DSD).

<u>General Systems Design (GSD)</u>:The GSD phase involves five primary activities: a) Analysis and initial translation of the business requirements, b) Identifying solution software and hardware components, c) Refinement and documentation of the systems technical requirements and preliminary design strategies into the systems requirements document (SRD), d) Evaluate and recommend solution alternatives (i.e., COTS, Transfer Technology, Custom, Cloud, etc), e) Creation of the GSD document that provides an overview of the conceptual models and diagrams (i.e., logical data, data/Information flows , work process flows, systems design and architecture frameworks, GUI, BI, reporting, etc) for the business solution. The finalized BRD and SRD are inputs for the Detailed Systems Design phase of the SDLC model. **<u>Note</u>**: A Requirement Specification Document (RSD) can be created that incorporates both the business and technical requirements (i.e., BRD and SRD) to form one comprehensive document for the major software release.

<u>Detailed Systems Design (DSD)</u>: The DSD phase involves three primary activities: a) Analysis and translation of the SRD and GSD, b) Detailed analysis, refinement, and documentation of the systems solutions software and hardware components, design and integration approaches, technical blueprints for both software and hardware

architectures and infrastructures), c) Creation of the DSD document that provides a comprehensive blueprint of the systems solution with detailed physical models and diagrams. The finalized DSD is the official blueprint from which the systems development teams can build/construct the business solution.

4) Coding/Construct Phase with Integrated Testing: On receiving system design documents, the work is divided in modules/units and actual coding is started. The system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality; this is referred to as Unit Testing. Units can be linked together to form modules that form major sub components of the system which are then merged/mapped together to form the final composite system (i.e., GUI, Database, Interfaces, Object Files, DLLs, logical and physical architectures, etc). Testing should be integrated into the code construct phase and performed at level of product development.

5) Testing: Software testing must be integrated into the code construct phase and executed at every level of product development. Comprehensive test plans must be established to ensure effective test coverage and effectiveness, detect and correct errors up stream, verify end product quality, and validate systems functionality and operational readiness. Test scenarios and their respective data sets must be derived and their correctness and consistency should be monitored throughout the development process with clear traceability to both functional and non-functional systems requirements. The multi level test strategy required throughout the Coding/Construct phase in order to realize a quality end product is as follows:
Hence the integrated multi-testing strategy will involve multiple but progressive testing levels:

a) Unit Testing: Unit testing mainly verifies coding constructs, techniques, standards compliance, and if the units/classes function properly and meet their specifications. Hence, each developer designs a unit level test and/or test case that invokes and exercises the methods/class validating proper execution, values, and expected outputs.

b) Module Testing: As specified above, the system is first divided into units which are developed and tested for their functionalities. These units are linked together into modules or sub systems components which are all assembled later to form a complete system. Hence, the module testing is to determine the proper operation of all subsystems and external subsystems interfaces, and execution paths and ensure they all work as the architect expected. The tests should be written and conducted by the senior developers of the development team. The end and results of this test level should determine if the code is ready to be fully assembled and migrated into the Systems Integration environment.

c) Systems Integration Testing (SIT): this testing is used to confirm that the entire system behaves as the user expects. Systems Integration tests should be derived from the user-level use cases which are designed to exercise and validate the code paths the users will execute. During this test phase software vulnerability testing as well as preliminary Load and Performance testing should be conducted. The end results of this test level should determine if the code is ready to migrate into the Pre-SAT and/or Systems Acceptance Testing (SAT) environment.

d) Pre-Systems Acceptance Testing (Pre-SAT): Pre-SAT environment(s) are functional staging environments within SAT environment(s) used by the development teams to conduct: 1) final Systems Integration tests and/or additional tests unable to be performed in SIT environment(s) due to the inherent constraints in the systems integration environment(s), 2) Ensure proper connectivity and proper operation of all systems components and interfaces, 3) Configure systems parameters in preparation for proper SAT execution.

e) Systems Acceptance Testing (SAT): SAT is the final testing phase that involves the end-users testing and validating of the production ready code. During test execution, the user actions are both structured and randomized to add to the realism of the testing and to better assess the reliability and response of the system. SAT is meant to validate compliance of both functional and non-functional requirements, systems proper operations, integration, and interoperability, and certify operational readiness to support live operational environments. The end results of this test level should determine if the code is ready to migrate into the Test for Production (TFP) environment(s).

f) Test For Production (TFP): This is the final staging area and stability period before the end product is deployed into live production environment(s).

Note: A comprehensive test plan document should outline the details of what, where, how, who, and when test phases of the integrated test and validation strategies along with the associated resource requirements are incorporated into the project timeline.

6) Implementation: It includes all the pre and post information systems solution deployment activities and support operations required to properly prepare, install, configure, and activate the information systems solution into live production environments from both technology (i.e., hardware/software platform) and business perspectives (i.e., end-user systems, process, and procedural training). The implementation playbook outlines the details for both the business and technical activities, validation checkpoints, and resources required for the implementation process while ensuring alignment with the agreed upon deployment strategy (i.e., pilot, phased roll-out, cut-over, big bang, etc).

7) Operations & Maintenance: This phase pertains to the life cycle management of this system and associated assets throughout its useful life. Business and technology drivers change overtime. Internal and external business demands influence program policy, operational, and legislative changes that often require business processes, procedures, and/or information systems to be modified or enhanced to accommodate these demands. From a technology perspective, technology assets are subject to useful life, standards compliance, and supportability constraints that require occasional information systems platform changes and/or migrations to remain viable supporting business operations. Lastly, problems with the information systems may surface which are not found during the systems development life cycle but after solution implementation. In this case, depending on the severity of the problem and priority it needs to be resolved; program change requests are submitted, reviewed, and typically bundled in the next scheduled software maintenance release.

II. **Software Development Process Models (SDPM):** There are various software development approaches defined that are used/employed during the development of custom software. These approaches are also referred to as Software Development Process Models (SDM). Each process model is a derivative of the SDLC used for the development of information systems that start from inception of the idea to final delivery of the business solution. The SDM approaches vary in the methods, processes, and project management style used to get from the initial concept to the final end product. In short, AGILE models are fluid/free flowing and highly iterative, loosely documented checklists, and very flexible unstructured methodologies as opposed to more traditional models that are very deliberate in structure and methodical, well documented, utilizing rigid gates. In some cases, there are derivatives within of each software development model and also hybrid combinations of both used depending on the project scope, complexity, and team composition and maturity, risk tolerance level, and the organizational structure and culture. BIS has defined two software development models: 1) Modified Water Fall (MWF) and 2) Rapid Application Development (RAD) derived from the SPIRAL model. The MWF is currently the predominant SDM used for all major systems modifications and enhancements whereas the RAD SDM is used internal to BIS for select web Internet and/or Intranet project initiatives requiring highly interactive and interactive customer involvement from concept to finished product with limited SEP and/or phase gate touch points.

III. **Capability Maturity Model:** A maturity model can be viewed as a set of predefined maturity levels that describe how well the behaviors, practices and processes of an organization can reliably and sustainably produce required outcomes. There are five levels defined along the continuum of the CMM and, according to the Software Engineering Institute (SEI): "Predictability, effectiveness, and control of an organization's software processes are believed to improve as the organization moves up these five levels. In short, the maturity levels range from ad-hoc (level 1) to optimizing (level 5). Within each of these maturity levels there are Key Process Areas (KPAs) which characterize that level and the stages that organizations must go through on the way to becoming mature. CMM process maturity can be developed incrementally from one level to the next and skipping levels is not allowed or feasible. When it is applied to an existing organization's software development processes, it allows an effective approach to assess the level of maturity, facilitating incremental process refinement and improvement. The model could be applied to other processes (ie. business processes and developing staff.) For software development processes, the CMM has been superseded by CMMI, though the CMM continues to be a general theoretical process capability model used in the public domain.

III. **Institute of Electrical and Electronics Engineering:** IEEE is a professional organization that provides an environment where members collaborate on technologies from computing engineering, software engineering and development, sustainable energy systems, aerospace, communications, robotics, healthcare, and more. DHS BIS leverages this resource along with other professional organizations to provide guidance in best practices as well as the development of software engineering and development guidelines and standards.

IV. Software Quality Assurance: Quality Assurance (QA)

Quality is determined by the product end users, clients and/or customers. Hence, SQA is used to establish a systematic monitoring and evaluation of the various aspects of the project, service or facility to maximize the probability that minimum standards of quality are being attained by the SDM processes. This refers to the processes used to create the deliverables. Examples of quality assurance include process checklists, SDM security, and/or project audits, and methodology and standards development. In addition it includes SDM testing activities (i.e., Integrated SDLC functional tests, load and performance, security vulnerability, integrity and validation, performance metrics and bench marking, etc). Hence, quality assurance activities are determined before production work begins and these activities are performed while the product is being developed. In contrast, Quality control activities are performed after the product is developed.

The DHS Modified Water Fall (MWF) Model partitions the software development lifecycle into seven major phases, as shown in Exhibit 2.0-1, Software Lifecycle Phases and Deliverables. Each phase is divided into activities and tasks, and has a measurable end (Phase Exit). The modified water fall model allows for iterative processes where feasible between phases.

The execution of all seven phases is based on the premise the quality and success of the software depends on:

- a feasible concept
- comprehensive and participatory project planning
- commitments to resources and schedules
- complete and accurate requirements
- a sound design
- consistent and maintainable programming techniques
- a comprehensive testing program.

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

A Structured Walkthrough is an organized procedure for reviewing and discussing the technical aspects of software development outputs (including documentation). For large projects, the project management team is required to perform at least one Structured Walkthrough for each lifecycle phase. The walkthrough is typically conducted by a group of peers; however, it usually includes reviewers outside of the developer's immediate peer group.

Depending upon organizational responsibilities or external audits, In-Phase Assessments (IPA) may also be performed. These assessments are independent reviews of outputs and deliverables developed (or revised) during each lifecycle phase.

Phase exits provide a way to review the outputs of each phase and determine whether all required artifacts are present. For large or highly critical projects, a phase exit takes place for each lifecycle phase.

The approval of the project sponsor and other project stakeholders at each phase exit enables both the project sponsor and project management team to remain in control of the project. It also prevents the project from proceeding beyond authorized milestones. The end results of the lifecycle are:

- the software
- data managed by the software
- associated documentation
- user training and support

The products and services are maintained throughout the project's lifecycle in accordance with documented configuration management procedures.

The MWF provides a method for performing the individual activities and tasks within an overall project framework. The phases and activities are designed to follow each other in an integrated fashion, regardless of whether the development phases are accomplished sequentially, concurrently, or cyclically.

Project teams, working with management, have the flexibility to adapt the SDLC to accommodate a particular software development process methodology (SDPM) (e.g., variations of Traditional, RAD, Agile techniques) with careful consideration given to business and technical team composition, competencies, maturity, experience, and other project constraints (e.g., federal mandates, risk tolerance, time to market, etc).

Note: Projects receiving federal or state funds must ensure all project plans and methodology deviations fall within the stipulations of legal and contractual obligations.

Documentation

Depending on the approved software development process model (SDPM), some documentation is created during specific phases and essentially remains unchanged throughout the systems development life cycle while others evolve continuously during the life cycle. Other documents are revised to reflect the results of analyses performed in later phases. Each of the documents produced are collected and stored in a project folder. Care should be taken, however, when processes are automated. Specifically, components are encouraged to incorporate a long-term retention and access policy for electronic processes. Be aware of legal concerns that implicate effectiveness of or impose restrictions on electronic data or records.

The amount of documentation required throughout the lifecycle depends on the approved SDPM driven by the scope, risk tolerance, and complexity of the project, as well as any stipulations tied to government funding. System documentation needs to be at a level allowing for full system operability, usability, and maintainability. Typically, projects requiring at least one work year of effort will have a full complement of documentation – smaller projects require less.

Note: Project security and quality assurance criteria may require the performance of other tasks along with the generation of additional documentation.

Exhibit 2.0-1 Software Lifecycle Phases and Deliverables

Feasibility Business Drivers Scope Definition High Level Requirements Business Case (Business and Technical perspectives) Executive Sponsorship Business Review Board Communities of Practice Project Initiation Charter and HLEs Initial Project Plan and scheduling All Phase Reviews	**Requirements Definition** Requirements Traceability Matrix (*initial*) Change Request Form Change Control Log Requirements Definition Document(RDD) Disaster Recovery Plan Description of Analysis Technique Test Plan (*initial, iterative revisions during build/construct and test phases*) Revise Detailed Project Plan and Schedule All Phase Reviews
General System Design Description of Design Technique General System Design (GSD) Document Requirements Traceability Matrix (*expanded)* System Architecture Document (*initial*) Capacity Plan (*initial*) Acquisition and Installation Plan Training Plan (*initial*) Conversion Plan (*initial*) Architecture Review Board Document All Phase Reviews	**Detailed System Design** Detailed System Design (DSD) Document Requirements Traceability Matrix (*expanded*) System Architecture Document (*revised*) Conversion Plan (*revised)* Electronic Commerce Security Assessment (ECSA) Document Test Plan (*revised)* All Phase Reviews
Software/Systems Development Application Code Unit Test Scenario Checklist Requirements Traceability Matrix (*expanded*) Unit/Module Test Plan Test Scenarios Operating Documentation Transition Plan (*initial*) Training Plan (*revised*) Capacity Plan (*final*) Batch Operations Manual Batch Operations Services Requests Test Plan (*final*) All Phase Reviews	**Software Integration and Testing** SIT Test Plan Test Report (System & Integration Test) Test Report (Load Test) Test Report (Regression Test) Test Report (Acceptance Test) Requirements Traceability Matrix (*final*) Operating Documentation (*revised)* Training Plan (*final*) All Phase Reviews
Acceptance and Installation SAT Test Plan Deployment Playbook Installation Test Materials Training Materials Operating Documentation (*final*) All Phase Reviews	**Operational Support Phase** Transition Plan (*revised*) All Phase Reviews

Note: A project may require deliverables in addition to those listed in this table.

2.1 Development Techniques

Applicability

The development team may pursue any one of a number of software development techniques as long as it allows for:

- adherence to applicable federal and state regulations
- meeting Advance Planning Document (APD)
- retention of fundamental software engineering objectives
- quality that has not been compromised

At times, an APD will already determine the development technique to be used on the project. In those situations, it is incumbent upon the project management team to work within the confines of the pre-determined development approach. When this is not the case, the project management team is free to explore viable alternatives.

Description

This section offers examples of development techniques compatible with the DHS methodology. The examples include high-level instructions on how to adapt the lifecycle phases to accommodate the development technique, but are not intended to be a comprehensive list of possible techniques.

Segmented Development

Large development projects may use a segmented development approach, which divides the requirements into functional segments. Each segment becomes a separate sub-project and provides a useful subset of the total capabilities of the full product.

This segmentation serves two purposes:

- The segments are more manageable, allowing project management and control tasks to be performed easier.
- Intermediate outputs become the building blocks for the complete product.

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

The development teams deliver segments to the project sponsor for evaluation or actual operation. Then they use the results to refine the content of the next segment, which provides additional capabilities. This process is repeated until the entire product has been developed. If significant problems are encountered with a segment, the developers may reexamine and revise the project objectives, modify the system architecture, update the overall schedule, or change how the segments are divided.

Two major advantages of this approach are:

- The project management team can demonstrate preliminary evidence the final product will work as specified

- Users will have access to, and use of, segments or functions prior to the delivery of the entire product

Spiral Development

Spiral development repeats the planning, requirements definition, and General System Design (GSD) phases in a succession of cycles. This allows the developers to clarify the project's objectives, define alternatives, identify risks and constraints, and construct a prototype. The prototype is evaluated and the next cycle is planned. Based on the evaluation, the developers refine project objectives, alternatives, constraints, and risks.

At that point, they may construct an improved prototype and repeat the process of refinement and prototyping as many times as necessary to provide a firm foundation on which to proceed with the project. The lifecycle activities for the Planning, Requirements Definition, and (GSD) Phases are repeated in each cycle. Once the design is firm, the development team follows the lifecycle phases for Detailed System Design (DSD), Development, and Software Integration and Testing to produce the final product.

Rapid Prototyping

Rapid prototyping can be applied to almost any software development technique. Software development projects using new technology or evolutionary requirements are likely candidates for rapid prototyping.

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

Users participate in an evaluation of the prototype, in order to refine the initial requirements and design. After confidence in the requirements and design approach is achieved, the final software is developed. The prototype might be discarded, or a portion of it may used to develop the final product.

The normal, software engineering documentation requirements are usually postponed due to prototyping efforts. Typically, the development team, project stakeholders, and project sponsor agree the prototype will be replaced with the actual software and required support documentation after proof of the model. The software replacing the prototype is developed using the lifecycle processes and activities.

Iterative Technique

The iterative technique is normally used to develop software piece by piece. Once the system architecture and GSD are defined and approved, system functionality can be divided into logically related pieces called "drivers." This technique enables progress to be visible earlier, and problems to be contained to a smaller scale.

The project management team iteratively performs DSD, code, unit test, and integration test activities for each driver, thereby delivering a working function of the product. These working functions or pieces of the software fit together as they are developed. This technique allows functions to be delivered incrementally for testing so they can work in parallel with the project management team. It also enables other functional areas, such as documentation and training, to begin performing their activities earlier and in parallel.

Rapid Application Development

Rapid Application Development (RAD) is a method for developing systems incrementally and delivering working pieces in short periods of time, rather than waiting until the entire product is constructed. RAD projects seek to avoid a common problem caused by long development intervals: the business has changed by the time the product is delivered.

RAD employs a variety of automated design and development tools, including Computer-Aided Software Engineering (CASE), fourth-generation languages (4GLs), visual programming, and graphical user interface (GUI) builders. RAD projects focus on personnel management and user involvement as much as on technology.

Joint Application Development

Joint Application Development (JAD) is a RAD concept involving cooperation between the designers of a computer system and the end users to develop a system to accurately meet the user's needs. It complements other system analysis and design techniques by emphasizing participative development among the project sponsors, users, designers, and builders. During DSD sessions, the system designer will take on the role of facilitator for meetings intended to address different design and deliverables issues.

Object-Oriented Development

Object-oriented (O.O.) development focuses on the design of software components that mimic the real world. An O.O. component that adequately mimics the real world is more likely to be reused, which is recognized as one of its more important advantages. Reused code has already been tested, and in the end, generally translates into cost savings. O.O. development may make code reuse much easier but, the amount of actual reuse still depends on the use of adequate repository tools and the motivation of the development staff.

Code reuse can also lead to faster software development. O.O. software is easier to maintain because its structure is inherently decoupled. This usually leads to fewer side effects when changes have to be made. In addition, O.O. systems may be easier to adapt and scale (i.e., large systems can be created by assembling reusable subsystems).

An O.O. project often follows an evolutionary spiral starting with customer communication, where the problem is defined. The technical work associated with the process follows the iterative path of analysis, design, programming, and testing. The fundamental core concepts in O.O. design involve the elements of classes, objects, and attributes. Understanding the definition and relationships of these elements is crucial in the application of O.O. technologies.

Issues needing to be understood before undertaking an O.O. project include:

- What are the basic concepts and principles applicable to O.O. thinking?
- How should O.O. software projects be planned and managed?
- What is O.O. analysis and how do its various models enable a software engineer to understand classes, relationships, and behavior?
- What is a ‘use case’ and how can it be applied to analyze the requirements of a system?
- How do conventional and O.O. approaches differ?
- What are the components of an O.O. design model?
- How are ‘patterns’ used in the creation of an O.O. design?
- What are the basic concepts and principles applicable for testing of O.O. software?
- How do testing strategies and test case design methods change when O.O. software is considered?
- What technical metrics are available for assessing the quality of O.O. software?

2.2 Commercial-Off-The-Shelf (COTS) Products Based Projects

Applicability

Many large-scale COTS integration projects will involve an Advance Planning Document (APD) including the rationale for using an off the shelf product.

Description

The trend in systems development is to make greater use of Commercial-Off-The-Shelf (COTS) products. This entails the purchase (and possible customization) of ready-made systems rather than in-house development "from scratch". This carries with it a sense of getting a system that can do the job, at a reasonable cost, and getting new functionality in subsequent releases over time.

This practice is often encouraged, and sometimes mandated by government agencies. There can be many benefits in using COTS products including improved quality and performance, speedier delivery of solutions, more cost-effective maintenance, and standardization across an organization. Additional benefits of a COTS solution may include:

- a proven track record of reliability
- known capabilities
- mitigated risk when feasibility testing can be arranged
- little or no custom programming, required for implementation

COTS and Open Systems

Many initiatives are under way in both private industry and government agencies to promote the use of an open systems approach, thereby anticipating even greater benefits than can be obtained from the use of COTS products alone. These initiatives are occurring because COTS products are not necessarily open and do not always conform to any recognized interface standards. Therefore, it is possible that using a COTS product commits the department to proprietary interfaces and solutions not common with any other product or system.

If the sole objective is the ability to capture new technology more cheaply, then the use of COTS products that are not "open" may satisfy requirements. However, considering the average COTS component is upgraded every 6 to 12 months, and new technology appears on the scene about every 18 to 24 months, money saved by procuring a COTS product with proprietary interfaces may quickly be lost in maintenance as products and interfaces change.

In the midst of all this, interface standards provide a source of stability. Without such standards, any change in the marketplace can impose an unanticipated change to systems using commonly found products.

NOTE: If business solution alternative recommendation is to use a COTS product, then the project team must use the process and procedures outlined in the DPW COTS evaluation and selection guideline.

COTS Planning Considerations

A COTS solution requires new and different investments including market research on available and emerging products and technologies, as well as COTS product evaluation and selection. The key to determining the best solution is to weigh the risks of straying from the following basic criteria:

- “Is it fully-defined?”
- “Is it available to the public?”
- “Is it maintained according to group consensus?”

Attention needs to be given to the following issues early in a project’s lifecycle:

- Market surveys to determine the availability of standards
- Selection of appropriate applicable standards
- Selection of standards-compliant implementations

Addressing these issues is necessary foundation for creating systems that serve current needs and yet can grow as technology advances and the marketplace changes. It is important for project teams to stay informed in this area, on an ongoing basis, with particular focus on:

- Upcoming revisions to specific standards
- Proposed changes in the new revision
- When ballots on the revisions are going to occur
- Where the implementations are headed

Skills Considerations

The depth of understanding, along with technical and management skills required on a project management team, is not necessarily diminished because of the use of COTS products. Arguably, the skills and understanding needed actually increase because of the following issues:

- Potential complexity of integration issues
- Consideration of long term system evolution as part of initial development
- The need to make informed decisions regarding products and standards

Types of COTS Solutions

COTS products can be applied to a full spectrum of system solutions, including (but not limited to) the following:

- Neatly packaged solutions such as Microsoft Office that require no integration with other components.
- COTS products that support the information management (e.g., Oracle or Sybase.) These systems typically consist of both COTS and customized components, with some “glue” code to enable them to work cooperatively.

- Systems comprised of both COTS and custom built products that provide large-scale functionality that are otherwise not available. These systems often require larger amounts of integration “glue” code.

COTS or Transfer technology Lifecycle Phases

All software development projects include planning, requirements definition, GSD, DSD, development, test, and software integration activities. The use of COTS products requires different lifecycle phases. The most fundamental change is the evaluation and selection of the COTS product, installation/integration, and system configuration and adoption into the business operations. The project management team will require a diverse project team with skilled business and systems analysts and systems architecture/engineering expertise to secure and successfully implement a COTS product.

This fundamental shift from custom development to COTS or transfer technology causes numerous technical, organizational, management, and business and technical challenges. Hence, requiring a modified SDLC with life cycle phases unique to a COTS solution. The COTS life cycle phases are defined as follows:

I. Business Feasibility & Research Phase:

1) Program defines business needs and works with BIS Portfolio managers to outline high level requirements, business case, establish priority, funding, and then develop a Charter.
2) With the approved charter, coordinate with BIS to further define/refine and understand requirements.
3) Conduct some preliminary research to identify plausible solution alternatives (COTS, Transfer Technologies, etc) and approaches.
4) Conduct an ARB I meeting to discuss the business needs and requirements and discuss potential solution approach and alternatives.

II. **Solicitation Phase:**

1) Solicit information directly to vendors using approved letter format and/or via RFI process with DGS based on the predefined business and technical requirements. Establish a final submission date for vendor responses. **Note**: This can be extended in the event responses are few or not as expected. In some cases, the solution alternatives have already been indentified or mandated but still require some level of inquiry.
2) Establish a listing of the potential candidates (e.g., COTS Vendors or Transfer technology considerations), direct contact, or other means).
3) Formulate and finalize the weighted evaluation matrix (M.S. Excel workbook template) to be used in the following phases.

III. **Preliminary Evaluation/Selection Phase:**

1) Form the multidisciplinary evaluation team.
2) Define detailed Business and technical requirements
3) Establish the preliminary evaluation worksheets to be used by the preliminary BTR evaluation team. This worksheet will be used to assist the evaluation team members in prequalifying vendors ensuring they meet the minimal business and technical requirements.
4) Conduct an initial review BTR to narrow the field identifying the best potential candidates. Evaluation team members shall review each vendor responses and complete and submit the preliminary evaluation worksheets for each vendor.

5) Summarize and validate results to determine the final list of qualified vendors who will be provided an opportunity to move on to Detailed Evaluation/Selection Phase, conducting a detailed review and evaluation and ARB II presentation base on BTR team preliminary evaluation results.
6) Finalize and send form letters to vendors participating in detailed evaluation/selection process (Phase IV).

IV. **Detailed Evaluation/Selection Phase**:
Establish the multidisciplinary evaluation team. Perhaps a broader team composition will/should be required extending to financial and business end users/SMEs. A diverse multidisciplinary evaluation team will be necessary to qualify vendors ensuring they completely satisfy or exceed all the business and technical requirements.

1) Establish the detailed evaluation worksheets to be used by the BTR evaluation team based on the approved weighted matrix. This worksheet will be used to assist the evaluation team members in qualifying vendors ensuring they completely satisfy or exceed all the key/critical business and technical requirements during the vendor product demonstrations.
2) Establish vendor demonstration meeting schedule based on vendor and BTR evaluation team and resource availability.
3) BTR team evaluates vendor demonstrations for each vendor to identify the best potential solution. Perform a detailed review and assessment for each solution alternative using the predefined worksheets and weighted matrix while participating in formal structured product demonstrations with the vendors. During the demonstrations, each evaluation team member shall assess, score, and document vendor responses (for their respective areas of expertise) using the evaluation worksheet documents for each vendor. Solicit further clarifying information from vendor if required.
4) Submit final worksheets to BIS project team lead.

V. **Assessment and Recommendations Phase**: Perform the comparative analysis, summarize results, and rank vendors. Record final results, outcomes, and recommendations in an Executive Summary Document for next phase. Finalize this document for executive management and ARB III review and approval.

1) Summarize and validate results to determine the final list of qualified vendors. Each BTR team member's worksheet results will be entered into the weighted evaluation matrix.
2) A comparative analysis will be conducted to qualify and rank vendors then summarize detailed evaluation results.
3) Evaluation team validates and forms consensus on final rankings and recommendations.
4) A draft Executive Summary Document (ESD) is created.
5) Conduct a meeting with TRT II detailed evaluation team members for their feedback and concurrences on the final draft of the ESD.
6) Finalize ESD for ARB III team review and concurrences.

VI. **Approval and Authorization Phase**: Present final Executive Summary Document to program sponsors/executive management and legal for their review and approval to move forward with final recommendations); this phase leverages ARB IV process.

VII. **Procurement Phase**: (Outline a detailed plan to procure and then operationalize the approved solution); this phase leverages ARB IV process.

VIII. **Implementation/Adoption Phase**

1) Implementing and integrating COTS products presents new challenges. A detailed implementation plan and playbook should be developed based on the deployment and adoption strategies from both business and technology perspectives. Although software COTS products are attempting to simulate the "plug and play" capability of the hardware world, in reality, they seldom plug into anything easily. Most products require some amount of systems configuration and integration to work harmoniously with other system components and environments. Secondly, the COTS platform may require data conversions and imports from existing systems and/or data sources.
2) COTS Adaptation does not imply customization of the COTS product but the configuration required to align with the business and technical operations. Adaptation can be a complex activity requiring business and technical expertise at the detailed system and specific COTS component levels. Adaptation and integration must take into account the possible interactions among custom components, COTS products, non-developmental item components, legacy code, and the architecture including infrastructure and middleware elements.

IX. **Testing & Validation**

Once the system has been installed, fully integrated and configured, there must be a determination of required testing levels. A COTS product usually functions as a "black box" and therefore changes the nature of testing. A system may use only a partial set of features of a given COTS product. Consequently, project designers must determine the number of COTS features to be tested. In other words, should a feature be tested if there are no plans to use the feature?

X. **Maintenance (LCM**)

Life cycle management and maintenance activities are vastly different in COTS-based systems development. The activities are not solely concerned with fixing existing behavior or incorporating new mission needs. Vendors update their COTS products on their schedules and at differing intervals. In addition, a vendor may elect to eliminate, change, add, or combine features for a subsequent release. Updates to one COTS product, such as new file formats or naming convention changes, can have unforeseen consequences for other COTS products in the system. To further complicate maintenance, all COTS products will require continual attention to license expirations and changes. All of these events routinely occur and may start before an organization installs the system or a major upgrade. The distinction between development and maintenance becomes less defined.

Adapting the SDM for COTS Projects

A modified SDLC can be used for COTS-based projects. The key is to follow the DHS COTS evaluation and selection guideline and elements of the SDM for identifying, securing, adapting, and implementing the COTS product using deliverables based project lifecycle to best suit the individual needs and characteristics of the project.

An example of lifecycle adaptation is offered in Exhibit 2.2-1, Example of COTS Life Cycle Projects. A project designer may combine multiple phases if the project will have a relatively small scope or short duration, and will use known technology. On the other hand, the traditional number of phases may be appropriate for large projects, with new technology, and long duration.

Documenting Deviations

Deviations from prescribed project deliverables must be documented with an explanation, and a statement which describes how project risk is not affected if a prescribed deliverable will not be produced.

> *Note: In some cases, especially with Express Projects, there may be an acknowledgement of increased risk due to tight schedules or the elimination of certain outputs. These risks must be documented and approved by DHS management.*
>
> *If the business solution alternative recommendation is to use a COTS product, then the project team must use the process and procedures outlined in the DHS COTS evaluation and selection guideline.*

Exhibit 2.2-1 Example of SDM Adapted for COTS Projects

COTS Phase	Project Phase	Deliverables
Business Feasibility Research	Project Initiation and Planning	Use DHS COTS Evaluation and Selection Guideline; COTS Project Plan (includes WBS)
Business Feasibility Research	Execution	COTS Project Plan (includes WBS)
Solicitation Phase	Execution	High Level Requirements
Preliminary Evaluation-Selection	Execution	Requirements Definition Document Systems Requirements Document Products to be Evaluated Architecture Review Board Document
Detailed Evaluation-Selection	Execution	Business and Technical Evaluation Matrix Architecture Review Board Document
Assessment-Recommendation	Execution	COTS Solution / Recommendation Draft COTS Executive Summary Document
Approval & Authorizations	Execution	Final COTS Executive Summary Document Architecture Review Board Document Sponsor Approval Letter
Procurements	Execution	Acquisition Plan
Implementation-Adoption		Transition Plan Implementation Playbook Operating Documentation Capacity Plan Conversion Plan Training Plan Systems Integration Test Plan & Report System Architecture Document Architecture Review Board Document Electronic Security Assessment Document
Testing-Validation	Requirements Definition	User Acceptance Test Plan Test Report
Life Cycle Management		LCM Governance Document

2.3 Quality Reviews

Applicability

The System Development Methodology (SDM) does not dictate department procedures, but provides the framework for their adherence.

Description

Quality reviews are performed to ensure the established system development and project management processes and procedures are being followed effectively – and exposures and risks to the current project plan are identified and addressed.

Quality reviews facilitate the early detection of problems affecting the reliability, maintainability, usability, availability, or security of the software. The four types of quality reviews are:

1. Peer Reviews
2. Structured Walkthroughs
3. In–Phase Assessments (IPA)
4. Phase Exits

The review to be used depends on the output being reviewed, the point of time within the phase, and the role of the person conducting the review.

Review Processes:

Peer Review

A peer review is an informal review of outputs (including documentation) conducted at any time, at the discretion of the developer. The developer's peers, who are typically other developers on the same project, attend these reviews.

Peer reviews can be held with relatively little preparation and follow-up activity. Review comments are informally collected, after which the product developer determines which comments require future action. The reviews, which focus on the specific content of a product, are designed to help the developer improve the product.

Some of the outputs prepared are considered interim, as they feed into a major deliverable or into another phase. Interim outputs are especially well suited to the peer review process, although all outputs are viable candidates. As always, the particular factors of each project help determine when peer reviews are conducted. Larger deliverables may go through multiple peer reviews to ensure they are free of defects.

Structured Walkthrough

The structured walkthrough is an organized procedure for reviewing and discussing the technical aspects of deliverables, including documentation. Like peer reviews, they are used to detect errors early in the development process, but are more formal in nature.

Various organizations have successfully used structured walkthroughs to identify errors in analysis, design, and requirements definition. In addition, walkthroughs have proven useful when validating the accuracy and completeness of deliverables.

Structured walkthroughs are conducted during all phases of the project lifecycle, after deliverables have been completed. They need to be scheduled in the Work Plan that was developed for the project plan and can be referred to as code reviews, design reviews, or inspections.

Structured walkthroughs can also be scheduled to review small, meaningful pieces of work, much like a peer review. The rate of progress for each lifecycle phase determines the frequency of the walkthroughs. However, they may be conducted more than once on a deliverable to ensure it is defect free.

In-Phase Assessment

The In-Phase Assessment (IPA) is a quality review usually conducted by an independent reviewer. The reviewer assesses the adherence to established standards and approved work plans. Also, software development and project management practices are reviewed to ensure sound development practices. This is particularly important when multiple deliverables are developed in a single lifecycle phase.

The reviewer assesses the deliverable and prepares an IPA report based on the content of the deliverable. An IPA does not require meetings among the involved parties to discuss the deliverable. However, the reviewer and the developer may meet after the IPA report is completed in order to review the findings. Subject matter experts (SME), such as documentation editors, may be used in addition to the assessor, to improve the quality of deliverables.

An IPA can be conducted anytime during a lifecycle phase when a deliverable is stable enough, or near the end of a phase to prepare for phase exit. A separate IPA can be conducted for each of the outputs, or one IPA for multiple outputs. The decision depends on the size of the outputs and their availability for review.

Phase Exit

The phase exit is conducted by the project management team with the project stakeholders, possibly consisting of:

- project sponsor
- user representative
- quality assurance personnel
- security point personnel
- architecture and standards representative
- development management

The phase exit is used to ensure the project meets DHS standards and an approved project plan. It is a high-level evaluation of all outputs developed in a lifecycle phase. The goal of the phase exit is to secure the concurrence of designated key individuals to continue with the project. This concurrence is a sign-off of the deliverables for the current phase of development including the updated project plan. It indicates all qualifications (issues and concerns) have been closed or have an acceptable plan for resolution.

At a phase exit meeting, the project management team communicates the following to the stakeholders and interested meeting participants:

- the positions of the key personnel
- any qualifications raised during the phase exit process
- issues remaining open from the IPA
- an action plan for resolution

The phase exit meeting is documented in summary form. Only one phase exit for each phase is necessary to obtain approval assuming all deliverables have been accepted as identified in the project plan

2.4 Gathering Metrics

Applicability

DHS has adopted a standard for service level objectives. Although not addressed during the initial development of the SDM, this important area needs to be considered during project planning.

Description

A Service Level Objective (SLO) is a contractual agreement for information technology services between the Department of Human Services (DHS) and a bureau, section, and/or division of DHS.

The SLO for gathering metrics compares metrics gathered on performance of on-line business, with established performance standards, for compliance with those standards. For additional information refer to Service Level Objective (SLO) for Gathering Metrics.

2.5 Information Management (IM) Reporting Considerations

Applicability

DHS is striving to mitigate project risk by introducing a mandatory reporting assessment early in a project's life cycle. The intent is to ensure that any IM reporting considerations are thoroughly discussed and carefully evaluated before the project has progressed too far along its development path. If it is determined that a project delivers an IM reporting component (e.g., Business Intelligence (BI), Data Warehouse (DW), or Operational Data Store (ODS)), then that project requires the adoption of IM standards and use of IM deliverable templates.

If the project in its entirety is an IM project, then the IM templates are to be used in the construction of deliverables. If a project contains a reporting component, then the IM templates are required only for that reporting sub-piece.

Description

The reporting assessment takes place before, and is reviewed during, Architecture Review Board 1 (ARB 1). In order to understand the reporting requirements of the project, questions are asked along the lines of those listed below:

- Is there a reporting component?

- Will data be stored in DW?
- Will there be reports, cubes, dashboards, scorecards, and/or maps?
- If not, why not?

Go to BIS Standards / Knowledge Management / IM Deliverables to view and access the IM deliverables matrix and IM deliverable templates). The matrix identifies the required deliverables.

Any questions, please contact BIS, Enterprise Knowledge Management Section.

3.0 Feasibility and Planning

Applicability

The planning phase of the SDM is critical to ensuring the project proceeds as approved during the Strategy phase of the Project Management Methodology. During this phase, a baseline assessment is done to verify the work statement for the project or the upcoming release aligns with the approved Community of Practice (CoP) template, Benefit Cost Analysis (BCA) template, Advance Planning Document (APD), and Program Revision Request (PRR). Any variances in the approved planning documents (CoP, BCA, APD, and PRR) must be identified and addressed prior to commencing systems development. These documents are all "living" documents and are subject to change throughout the life of the project. Because of this, the submittal of updated or revised document(s) may likely be required to request confirmed approval of a project in its current state.

Description

In the most general of terms, the purpose of the SDM Planning Phase is to verify the project is being developed only as approved. Referring back to the output documents from the Project Management Methodology planning processes serves the additional purpose of ensuring those processes were followed and the project is indeed funded and approved by the necessary State and Federal entities before moving forward. If these processes were required and not followed early in the project life cycle, project activity will be delayed until the project is formally approved and funded.

In this first phase of software development, the following activities are executed, as appropriate:

- The approved CoP is referenced and compared to the project's scope
- The approved BCA is referenced and compared to the project's budget
- The approved APD is referenced and compared to the current project status relative to project schedule, cost, and concept
- The approved PRR is referenced and validated
- The approved budget is compared to the project budget to ensure it does not exceed the funded amount

If the above-mentioned planning documents require revisions, these are made and the documents are resubmitted to the appropriate entity(ies)

Inputs

The following items provide input to this phase (as applicable):

- Approved Community of Practice Template
- Approved Benefit Cost Analysis Template
- Approved Advance Planning Document
- Approved Program Revision Request
- Approved Budget

Outputs

The following outputs to this phase include (as applicable):

- Revised Community of Practice Template
- Revised Benefit Cost Analysis Template
- Advance Planning Document Update
- Revised Program Revision Request

4.0 Requirements Definition

Applicability

The relevance of this phase for Department of Human Services (DHS) development is similar to the previous phase. For many medium and large projects DHS development teams undertake, the Information Resource Management (IRM) Cross-Program Team coordinates the requirements definition activities.

At times, the IRM Cross-Program team may request input from DHS application developers. In those cases, the activities are subject to processes defined by the coordinating offices. Therefore, the activities specified in this phase apply to projects where the DHS development teams coordinate all requirements tasks – or when externally defined requirements are supplemented.

Description

The primary goal of this phase is to develop mutual understanding among the project sponsor, the users, and the project management team as to what the project requirements are. The requirements are then documented. These requirements result in an approved and complete Requirements Definition Document (RDD). The RDD becomes the initial baseline for design and a reference for determining whether the completed project meets approved requirements.

The Requirements Definition Phase includes:

- Analysis of the users' business processes and needs
- Translation of processes and needs into formal requirements
- Planning of the testing activities to validate the performance of the product

> *Note: The testing activities of all projects, regardless of size, always need to be based upon the requirements agreed upon in this phase.*

Inputs

Inputs include the:

- Scope
- Charter
- All output from the planning phase (this includes, but, is not limited to the documented goals and objectives for the project, copies of management directives, legislative guidelines, and any use case scenarios already established)
- The Project Plan
- Data Processing Service Request (DPSR) and all associated attachments

High-Level Activities

This phase is divided into high-level activities performed during the phase. These activities represent the minimum requirements for a large software development effort. Notes are provided to assist customizing the lifecycle phase requirements to accommodate project scalability. High-level activities for the Requirements Definition Phase consist of:

4.1 Manage Requirements

4.2 Select Requirements Analysis Technique

4.3 Define Project Requirements

4.4 Define Data Backup/Restore and Disaster Recovery Requirements

4.5 Define Data Requirements

4.6 Create Initial Logical Data Model

4.7 Compile and Document Project Requirements

4.8 Develop Test Plan

4.9 Conduct Phase Review

Outputs

Deviations in content and delivery are dependent upon the size and complexity of a project. The outputs of a typical large software project consist of:

- Requirements Traceability Matrix
- Change Request Form
- Change Control Log
- Requirements Definition Document (RDD)
- Disaster Recovery Plan
- Description of Analysis Technique
- Test Plan

A matrix showing the outputs associated with each high-level activity is provided in Exhibit 4.0-1, Requirements Definition High Level Activities and Outputs.

Review Process

It is important to perform quality reviews to validate the outputs. The activities appropriate for quality reviews are identified in this phase and 2.3 Quality Reviews. The time and resources needed to conduct the quality reviews must be consistent with the project resources, schedule, and Work Plan Standard.

Reference/Resource

Section 2.3 Quality Reviews, provides an overview of the types of reviews to be conducted within a project.

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

Checklist

Requirements Definition Phase Checklist

Exhibit 4.0-1 Requirements Definition High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
4.1	Manage Requirements	Requirements Traceability Matrix (*initial*) Project Change Request (PCR) Form	Y N
4.2	Select Requirements Analysis Technique	Description of Analysis Technique	N
4.3	Define Project Requirements	Requirements Definition Document (Process Model Narratives)	Y
4.4	Define Data Backup/Restore and Disaster Recovery Requirements	Data Backup/Restore Requirements Disaster Recovery Plan	Y Y
4.5	Define Data Requirements	Requirements Definition Document (Data Dictionary)	Y
4.6	Create Initial Logical Data Model	Requirements Definition Document (Logical Data Model)	Y
4.7	Compile and Document Project Requirements	Requirements Definition Document	Y
4.8	Develop Test Plan	Test Plan (*initial*)	Y
4.9	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is an output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

4.1 Manage Requirements

Responsibility

Project Management Team

Applicability

Team members gather, organize, prioritize, and document project requirements. Also, verification of all requirements is captured and any changes are tracked to the initial requirements.

Description

Team members gather, organize, verify, prioritize, and document project requirements. Any changes are tracked to the initial requirements.

Requirements Management is the process of developing initial requirements for a project, evaluating and approving any proposed changes, and managing the various outputs affected by changes to the requirements. Team members document the needs and expectations, reach an agreement about which requirements to address for the current project, and which to defer or eliminate.

Identify all requirements in the Requirements Definition Document (RDD) as specified in activity 4.3 Define Project Requirements. As the project progresses, identify and manage additional requirements through the change control process. Larger projects are required to use a Requirements Traceability Matrix, which uniquely identifies each requirement in the RDD. The matrix is a tool ensuring requirements are traced and verified through the various lifecycle phases.

Trace the requirements from external sources, such as the user, through the derived system-level requirements, all the way to specific hardware/software project requirements. All requirements must be cross-traceable to design, implementation, and test artifacts ensuring requirements are satisfied.

Establish how changes to the original project scope will be handled by following the Project Change Management process located in task 4.1.2 Manage Project Change.

Outputs

Develop the Requirements Traceability Matrix to verify requirements are met and the project remains within scope. Additional tasks contain information on additional outputs. Maintain the documentation in the Project Folder.

Review Process

Review the outputs for accuracy and completeness and maintain the document in the Project Folder.

Template

Requirements Traceability Matrix

Tasks

The following tasks are involved in requirements management:

4.1.1 Develop a Requirements Traceability Matrix

4.1.2 Manage Project Change

4.1.1 Develop a Requirements Traceability Matrix

Responsibility

Project Management Team

Applicability

The Requirements Traceability is used to track requirements throughout the lifecycle of the project.

Description

A Requirements Traceability Matrix is a tool constructed to ensure all confirmed requirements are accounted for in the statement of work, in the solicitation, in the vendor response, in the contract, and in the deliverables themselves.

All outputs developed in subsequent lifecycle phases are traceable back to the project requirements described in the Requirements Definition Document (RDD), just as every project requirement must be traceable to a specific project objective described in the Project Plan. This traceability ensures the project will satisfy all the requirements and remain within scope, with no inappropriate or extraneous functionality.

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

Outputs

A matrix is developed to trace requirements back to the project objectives identified in the Project Plan. It is updated throughout the lifecycle of the project and maintained in the Project Folder. The matrix contains the following fields:

- A unique identification number containing the general category of the requirement (e.g. SYSADM) and a number assigned in ascending order (e.g. 1.0; 1.1; 1.2)
- The requirement name
- Requirements Definition ID number containing the requirement
- General System Design (GSD) ID number containing the requirement
- Detailed System Design (DSD) ID number containing the requirement
- Program Module/Application Object containing the requirement
- Test Script containing the requirement test
- Test Result ID number(s) where requirement was tested
- Modification field. If the requirement was changed, eliminated, or replaced, indicate disposition and authority for this modification
- Comments

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix to ensure all the requirements have been accurately captured.

Sample Traceability Matrix

One method for tracing requirements is a threading matrix, grouping requirements by project objectives. Under each project objective, the source of the requirement, the unique requirement identification number, and the lifecycle activities are listed in columns along the top and the project requirements in rows along the left side. As the project progresses through the lifecycle phases, a reference to each requirement is entered in the cell corresponding to the appropriate lifecycle activity.

Template

Requirements Traceability Matrix

Reference/Resource

Structured Walkthrough Process Guide

4.1.2 Manage Project Change

Responsibility

Project Management Team

Applicability

Effective project management requires disciplined control of changes to project scope. Even seemingly simple changes can have subtle yet very serious consequences if not managed correctly. It is intended for use by DHS project leads, members of the Change Control Board, the Project Management Team, the project's Steering Team, DHS's financial community members, and project stakeholders.

Description

Project Change Management is the formal process for making changes to the project's original scope. A project begins with a well-documented project scope agreed to by both the Project Management Team and the end-user. It generally involves redefining existing objectives and deliverables or specifying new project objectives and deliverables. Having a standardized, disciplined project change procedure ensures changes affecting a project's baseline are accepted and implemented only after a thorough assessment is made of the impact.

Outputs

Maintain the Project Change Request (PCR) form in the Project Folder.

Review Process

Conduct a peer review or structured walkthrough on the PCR when a significant change is requested.

Reference/Resource

Structured Walkthrough Process Guide

Project Change Management Guideline

Project Change Management Process Map

4.2 Select Requirements Analysis Technique

Responsibility

Project Management Team

Applicability

The Department of Human Services (DHS) Development team is free to choose an analysis technique fitting both the technical and cultural aspects of the project providing it adheres to all relevant standards. The System Development Methodology (SDM) does not mandate any single technique, but recommends the decision be based on industry best practice, experience of the team members, and department guidelines.

Description

A requirements analysis technique is a set of data collection and analysis techniques (e.g. user interviews and rapid prototyping) combined with the lifecycle requirements used to identify the software requirements.

The selected requirements analysis technique needs to be congruous with the:

- Type, size, and scope of the project
- Number, location, and technical expertise of the users
- Anticipated level of involvement of the users in the data collection and analysis processes

The requirements analysis technique ensures the functionality, the performance expectations, and the constraints of the project are accurately identified from the users' perspective. In addition, the analysis of requirements mitigate the potential impact on existing operations and business practices, future maintenance activities, and the ability to support the project sponsor's long-range information resource management plans.

The technique is repeatable (for similar projects), therefore, allowing the project management team and users to become familiar and comfortable with the selected approach. Discuss the analysis technique with the project sponsor and users to make sure the process being used, their role and responsibilities, and the format of the output (i.e., how the requirements will be organized and described) are understood.

Outputs

A description of the analysis technique is distributed to members of the development team, project sponsor, and users involved in the requirements definition process. Maintain the document in the Project Folder.

Review Process

Conduct a structured walkthrough of the analysis technique to verify appropriateness for the project. A walkthrough is not required if the technique has been used successfully on similar projects, with the same development group.

Reference/Resource

Structured Walkthrough Process Guide

4.3 Define Project Requirements

Responsibility

Project Management Team

Applicability

Requirements drive the development process from the initial steps to the final solution.

Description

The team uses the project scope, objectives, and high-level requirements as the basis for defining the project requirements. Questions used to define the project objectives are helpful in developing the requirements.

The goal is to identify *"what functions are to be performed on what data, to produce what results, at what location, and for whom."*

Avoid incorporating design issues and specifications in the requirements. One of the most difficult tasks is determining the difference between *"what"* is required and *"how to"* accomplish what is required. Generally, a requirement specifies an externally visible function or attribute of a system – *"what".* A design describes a particular instance of how the visible function or attribute can be achieved – *"how to".*

Requirements must support the project sponsor's business needs and long-range plans, as well as those of the Department of Human Services (DHS) and the Commonwealth. Specify and document requirements as completely and thoroughly as possible. Requirements not documented are not considered to be approved.

Attributes

Prior to the delivery of the Requirements Definition Document (RDD), review each of the requirements to ensure the following attributes:

- *Necessary* – Absolute requirements verified are indicated by "must" or "shall". Goals or intended functionality are indicated by "will".
- *Correct* – Each requirement is an accurate description of a feature or process of the project.
- *Unambiguous* – The statement of each requirement denotes only one interpretation.
- *Complete* – Each requirement describes one result to be achieved by the product. The requirement does not describe the means of obtaining the result.
- *Consistent* – Individual requirements are not in conflict with other requirements.
- *Verifiable* (testable) – Each requirement is stated in concrete terms and measurable quantities. A process needs to be in place to validate the satisfaction of the set of requirements.
- *Modifiable* – Any necessary changes to the structure and style of the requirements are complete, consistent, and as simple as possible.
- *Traceable* – The origin of each requirement is clear and tracked in future development activities and tests.

Identification System

Use a standard identification system for all requirements to facilitate configuration control, requirements traceability, and testing activities. Select an identification system fitting the requirements' analysis technique and the project scope.

The identification system must provide a unique designator for each requirement. For example, the identification system can classify the requirements by type (e.g. functional, input, or computer security). Within each type classification, the requirements are assigned a sequential number.

Changes

As the project proceeds, the requirements may change or expand to reflect modifications in the users' business plans, design considerations and constraints, advances in technology, and increased insight into user business processes. Use a formal change control process to identify, control, track, and report proposed and approved changes. For additional information, refer to task 4.1.2 Manage Project Change.

Incorporate approved changes in the RDD to provide an accurate and complete audit trail of the changes.

Outputs

Requirements Definition Document

Review Process

Conduct a peer review or structured walkthrough on the RDD.

Template

Requirements Definition Document

Reference/Resource

Structured Walkthrough Process Guide

Tasks

Defining project requirements includes the following tasks:

4.3.1 Define Process Model Narrative

4.3.2 Define Security Requirements

4.3.3 Define Batch Job Requirements

4.3.4 Define Workflow Requirements

4.3.5 Define External Interface Requirements

4.3.6 Define Printing Requirements

4.3.7 Define Imaging Requirements

4.3.8 Define Reporting Requirements

4.3.9 Define Business Rule Requirements

4.3.10 Define System Availability Requirements

4.3.11 Define Performance and Capacity Requirements

4.3.12 Define Conversion Requirements

4.3.1 Define Process Model Narrative

Responsibility

Project Management Team

Applicability

Requirements drive the development process from the initial steps to the final solution. The process model narrative includes a completed description of what the process is attempting to accomplish and any relevant background information.

Description

Requirements for all functions (automated or manual) must be identified, including a description of automated and manual inputs, processing, outputs, and conditions for all functions. These requirements become the process model narrative and are broken down into activities for the functionality to be successful.

Outputs

A record of the process model narrative is maintained in the Project Folder and incorporated into the Requirements Definition Document (RDD).

Review Process

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

Reference/Resource

Structured Walkthrough Process Guide

4.3.2 Define Security Requirements

Responsibility

Project Management Team

Applicability

The Governor's Office of Administration/Office for Information Technology (OA/OIT) Security Policies has security standards and procedures in place for all Commonwealth agencies. In addition, verify any federal restrictions.

Description

Develop the security requirements in conjunction with any Commonwealth standards. This early involvement determines the classification and the level of access required for the software. Establish appropriate safeguards to protect sensitive information from accidental disclosure.

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

Sample Access Control Questions

A list of sample questions to help define the access controls for the software includes:

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

Outputs

The security and access requirements are defined in the Security Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).Review Process

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

Example

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

4.3.3 Define Batch Job Requirements

Responsibility

Project Management Team

Applicability

If the system generates any batch jobs, they must be documented within this task.

Description

Capture the batch job requirements for each process. If there are no batch job requirements for a particular process or for the whole application, this task can be omitted. A batch job is a set of programs executed by the operating system as a single unit. During the batch job, the user does not interact with the system. In interactive processing, the user communicates with the computer while the program is running, perhaps giving instructions for each item.

Outputs

The batch job requirements are defined in the Batch Job section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

Batch Job Requirement Example
The application will reconcile all paid accounts for the day and will be run daily on weekdays.

4.3.4 Define Workflow Requirements

Responsibility

Project Management Team

Applicability

If the system generates a workflow, it must be documented within this task.

Description

Workflow refers to the path an object takes during a review/approval process. It is literally the flow of work through a process. If there are no workflow requirements for a particular process or for the whole application, this task can be omitted.

Outputs

The workflow requirements are defined in the Workflow Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

Workflow Requirement Example
Upon submittal of the personnel account information, the item will be placed in the workflow queue for the department manager to review. If the manager clears the item, the personnel account information will finish processing.

4.3.5 Define External Interface Requirements

Responsibility

Project Management Team

Applicability

The hardware and software interface requirements must specify the interfaces required in supporting the development, operation, and maintenance of the software.

Description

The hardware and software interface requirements must specify the interfaces required in supporting the development, operation, and maintenance of the software. When defining the system interface requirements, consider the:

- Project sponsor's and users' computing environment
- Existing or planned software providing or accepting data from the software
- Other organizations or users having or needing access to the software
- Purpose or mission of interfacing software
- Common users, data elements, reports, and sources for forms/events/outputs
- Timing considerations influencing the sharing of data, direction of data exchange, and security constraints
- Development constraints such as the operating system, database management system, language compiler, tools, utilities, and network protocol drivers
- Standardized system architecture defined by hardware and software configurations for the affected organizations, program offices, sites, or telecommunications programs

If there are no external interface requirements for a particular process or for the whole application, this task can be omitted.

Outputs

The external interface requirements are defined in the External Interface Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

External Interface Requirement Example
The application must maintain a record of all user access attempts sorted by authorized and unauthorized users.

4.3.6 Define Printing Requirements

Responsibility

Project Management Team

Applicability

If the system is to produce printed output it must be documented within this task.

Description

Printing requirements involve the formatting and printing of anything generated from the screen. This includes reports, receipts, or anything else requiring a hard copy. If there are no printing requirements for a particular process or for the whole application, this task can be omitted.

Outputs

The printing requirements are defined in the Printing Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

Printing Requirement Example
The application will allow the account information to be printed by the user with all of the information except their SSN.

4.3.7 Define Imaging Requirements

Responsibility

Project Management Team

Applicability

If the system is to incorporate any scanning or imaging technologies, it must be documented within this task.

Description

Imaging is capturing any document, file, or physical item stored in the system. If there are no imaging requirements for a particular process or for the whole application, this task can be omitted.

Outputs

The imaging requirements are defined in the Imaging Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

Imaging Requirement Example
The application will allow users' voided checks to be uploaded into the system to store as part of their user profiles.

4.3.8 Define Reporting Requirements

Responsibility

Project Management Team

Applicability

If the system is to produce reports they must be documented within this task.

Description

Reporting is the generation of reports based on data or statistics from past processes or events of the system. If there are no reporting requirements for a particular process or for the whole application, this task can be omitted.

Outputs

The reporting requirements are defined in the Reporting Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

Reporting Requirement Example
All of the payments processed in one day will be used to populate a daily report summary consisting of the payment ID, payment amount, payee, and check number.

4.3.9 Define Business Rule Requirements

Responsibility

Project Management Team

Applicability

If there is business rule software used for the application, it must be documented within this task.

Business rules will be required for the Physical Data Model in the Detailed System Design (DSD) phase.

Description

A business rule defines or constrains one aspect of the business intended to assert business structure or influence the behavior of the business. In essence, it is a business functionality conforming to an "if-then" statement. A good business rule is cohesive: in other words, it describes one, and only one, concept.   By ensuring business rules are cohesive, they are easier to define and increase the likelihood they will be reused (every time one of the artifacts refers to a business rule, even other business rules, it is effectively being reused). The business rule differs from normal business functions by being captured for use in the external Business Rules Engine software. If there is no business rule software to be used for this application, this task can be omitted.

Outputs

The business rule requirements are defined in the Business Rule Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

Business Rule Requirement Example
If the user's account information matches the reconciliation criteria, then the user's account will be marked as "reconciled".

4.3.10 Define System Availability Requirements

Responsibility

Project Management Team

Applicability

The availability of the system needs to be mentioned in this section (for example 24/7/365, or during normal business hours (8am-5pm).

Description

System availability requirements define when certain parts of the application need to be available for operation usage. The information gathered in defining the project objectives can translate into very specific availability requirements (e.g. if work performed for an organization is mission critical to the Department, the hours of operation and throughput will be critical to meeting the mission). In addition, federal, state, or department policy can dictate specific availability times.

Outputs

The system availability requirements are defined in the System Availability Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

System Availability Requirement Example
The application must be able to handle personnel account data entry from 9am to 5pm Monday through Friday.

4.3.11 Define Performance and Capacity Requirements

Responsibility

Project Management Team

Applicability

Identify all performance requirements the system is expected to meet. Include performance-related characteristics, such as online response time, batch turnaround time, capacity limits, accuracy and validity requirements, input and output effects on performance, and failure contingencies.

Description

Performance and capacity requirements define how the software must function (e.g. response times and throughput under various load conditions). The information gathered in defining the project objectives can translate into very specific performance requirements (e.g. if work performed for an organization is mission critical to the Department, the performance and process transactions will be critical to meeting the mission). In addition, federal, state, or department policy can dictate specific response times.

Outputs

The performance and capacity requirements are defined in the Performance and Capacity Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

Structured walkthroughs are conducted, as deemed necessary, to ensure the necessity, testability, accuracy, and completeness of the performance requirements.

Example

Performance Requirement Example
The application must be able to handle 10,000 data entry transactions per day.

4.3.12 Define Conversion Requirements

Responsibility

Project Management Team

Applicability

If the new system being developed is replacing an existing system, the requirements for converting the data into the new format must be defined.

Description

Describe the data to be converted from existing sources being used by the application.

Outputs

The conversion requirements are defined in the Conversion Requirements section of the Process Model Narrative. Incorporate these requirements into the Requirements Definition Document (RDD).

Review Process

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

Example

Conversion Requirement Example
The existing account codes from the HRIS system and relevant columns need to be converted into this application.

4.4 Define Data Backup/Restore and Disaster Recovery Requirements

Responsibility

Project Management Team

Applicability

All critical systems must have a Data Backup/Restore and a Disaster Recovery plan developed to ensure continuous service to the customer. Small or express projects creating non-critical products need to have backup procedures defined to ensure recovery should the need arise.

Description

Generate requirements for data backup and restore recovery. If the system is defined as mission critical, provide a Disaster Recovery Plan for operational startup in conjunction with the site authority in the event of a declared emergency.

Outputs

The Data Backup and Restore Requirement's Checklist is required for all applications.

The Disaster Recovery Plan includes all mission critical systems. Provide a Continuity of Operations (COOP) Statement for systems not mission critical.

Review Process

Conduct structured walkthroughs, as needed, to ensure the necessity, testability, accuracy, and completeness of the disaster recovery requirements.

Template

Disaster Recovery Plan

Checklist

Data Backup and Restore Requirement's Checklist

Reference/Resource

Structured Walkthrough Process Guide

OIT Continuity of Government (CoG)

4.5 Define Data Requirements

Responsibility

Project Management Team

Applicability

All data elements serving as inputs and/or outputs must be defined and documented in the data dictionary.

Description

Identify the data elements to be accessed, stored, and processed by the application. This process begins during the Requirements Definition Phase and is expanded in subsequent phases.

Data requirements may include the need for data purification, either as a conversion effort or as an ongoing activity, and additional issues such as data archival and purging. Identify the initial conversion efforts in the Process Model Narrative and update in subsequent phases.

Outputs

The major output of the data requirements identification task is a Data Dictionary. A data dictionary provides an ordered set of definitions about inputs, outputs, and data storage. In the Requirements Definition Phase, the data dictionary contains a minimum amount of information about the data elements. This includes the definitions of the entities, how the data is stored, and how the data flows to or from other applications. The Data Dictionary is refined during the design phases as the data elements are documented in more detail, and the logical groupings of data elements are formed into interrelated tables or record descriptions.

Maintain the data dictionary in the Project Folder. The data dictionary is used as input into the Requirements Definition Document (RDD).

Review Process

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

Reference/Resource

Structured Walkthrough Process Guide

Instructions for Completing the Data Dictionary Excel Spreadsheet

4.6 Create Initial Logical Data Model

Responsibility

Project Management Team

Applicability

This activity is typically performed by projects incorporating structured (i.e., functional decomposition) methods. Object-based projects include an analysis model, but do not develop conventional data flow diagrams.

Description

The logical data model defines the flow of data through the software system and determines a logically consistent structure for the software. Each module defining a function is identified, interfaces between modules are established, and design constraints and limitations are described. The logical data model focuses on the real world problem to be solved.

The characteristics of a logical data model are:

- To describe the final sources and destinations of data and control flows crossing the system boundary (as opposed to intermediate handlers of the flows)
- To describe the net transfer of data across the system boundary rather than the details of the data transfer
- To provide for data stores only when required by an externally imposed time delay

When creating a logical data model, the organization of the model must follow the natural organization of the subject matter. The names given to the components of the model must be specific. The connections among the components of the model need to be as simple as possible.

Outputs

The logical data model is documented in user terminology and contains sufficient detail to obtain the project sponsor's and users' approval. Use the data flow diagrams to show the level of detail needed to reach a clear, complete picture of processes, data flow, and data stores.

Maintain the logical data model and data flow diagrams in the Project Folder and include them in the Requirements Definition Document (RDD). They will be modified throughout the length of the project in the subsequent General System Design (GSD) and Detailed System Design (DSD) Phases.

Review Process

Conduct a structured walkthrough to verify the logical data model is correct, logical, and complete.

References/Resources

Structured Walkthrough Process Guide

ITP-INF003 Data Modeling Standards

4.7 Compile and Develop Requirements Definition Document (RDD)

Responsibility

Project Management Team

Applicability

All projects must include a requirements definition except express projects and small projects where the initial request details adequately the requirements.

Description

Compile the requirements gathered during the requirements analysis process in preparation for development of the initial Requirements Definition Document (RDD). Requirements compilation includes:

- Specify the logical requirements and data requirements without dictating a physical design or technical solutions
- Write the requirements in non-technical language understandable by the project sponsor and users
- Organize the requirements into meaningful groupings as structured in the RDD template
- Develop a numbering scheme for the unique identification of each requirement
- Select a method for:
    - Tracing the requirements back to the sources of information (Requirements Traceability Matrix)
    - Threading requirements through all subsequent lifecycle activities

The RDD contains the business functionality requirements captured in a specific format. Emphasis is placed on specifying functionality without describing how the functions are provided. The *"how"* of the implementation will be specified in the design phase.

Note: A project describes a particular sequence of activities and associated resources defining a process to produce the software. Project information is not to be included in the RDD. However, certain information may be considered a requirement in one project but design or implementation details in another project.

Outputs

The RDD integrates and documents the requirements gathered during the Requirements Definition Phase, such as the Process Model Narrative, Data Dictionary, and Logical Data Model.

Review Process

Conduct structured walkthroughs, as needed, to ensure the RDD is accurate, complete, and express the requirements in a manner understood by the project sponsor.

The completion of the initial RDD is an appropriate time to schedule an In-Phase Assessment (IPA).

Template

Requirements Definition Document

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

4.8 Develop Test Plan

Responsibility

Project Management Team

Applicability

For small and express development projects, a formal Test Plan may not be necessary. In those cases, a statement of the test approach is sufficient. Acceptance testing (task 4.8.4 Develop User Acceptance Test Plan) is discussed separately in more detail.

> *Note: Planned testing is required for all projects, regardless of size and complexity.*

Description

The Test Plan is a narrative and tabular description of the test activities planned for the project. The Test Plan establishes the systematic testing necessary to ensure the project meets all requirements and the deliverables are in accordance with existing standards.

The test plan includes the resources, team responsibilities, and management techniques needed for planning, developing, and implementing the testing activities occurring throughout the lifecycle. The plan includes the roles and responsibilities of any external test groups participating in the testing. The project management team is responsible for the coordination with external groups.

At a high level, the plan focuses on identifying the testing techniques and phases. The plan includes detailed information about testing (i.e., test plans, procedures, and reports) as the project progresses.

The project sponsor must review and approve the Project Test Plan prior to conducting any tests.

Inputs

Inputs include:

- The Project Plan
- The Requirements Definition Document (RDD)
- The statement of goals and objective for the project

Outputs

The test plan must include as much detail as possible. However, some details may not be fully specified until the Detailed System Design (DSD) is complete. A completed Test Plan includes:

- A description of the occurrence and timing of the test phases in the lifecycle along with entrance and exit criteria for each test phase
- A specification of the test products at each test phase, including a description of the types of testing activities to be performed, test cycle, durations and criteria for pass/fail

- A mapping of what requirements are verified in what test phase
- Criteria for evaluating the test results of each test phase including the users involved
- An initial estimate of the resources necessary to accomplish the testing
- Identification of the appropriate person or group to conduct each type of testing activity
- An outline of the test environment (hardware, software, test tools, data, and location) needed to conduct the tests
- A preliminary schedule for executing the test activities

Review Process

Conduct a structured walkthrough to ensure the Project Test Plan adequately describes all testing activities, test schedules, test products, test responsibilities, the testing methodology, and the required resources. Include the project sponsor, primary users, and other relevant stakeholders.

Template

Project Test Plan

Reference/Resource

Structured Walkthrough Process Guide

Tasks

Preparation of the Project Test Plan involves the following tasks:

4.8.1 Identify Test Techniques

4.8.2 Identify Test Phases

4.8.3 Identify Test Environment Requirements

4.8.4 Develop User Acceptance Test Plan

4.8.1 Identify Test Techniques

Responsibility

Project Management Team

Applicability

This task is to be followed for:

- All new and large projects
- Where the project teams are not experienced with the software, platform, or business domains involved
- Where there are more than two or three development teams

For smaller projects where the development is considered routine and generally is known to follow best practices this task can be omitted.

Description

The use of a standard testing methodology or automated tool may affect or dictate the selection of test techniques to be applied. The testing may be performed by an independent test group. In this situation, the technique to be used might be specified in the contract, otherwise, the contractor supplies documentation to adequately describe the techniques to be used so they are repeatable and understood. If there is not an independent group, then the development team is required to use any appropriate Commonwealth standards currently in place.

Any exceptions to the standards must be specified in the Test Plan.

If no testing standards are in place, the team selects an appropriate technique.

The Test Plan specifies the testing techniques planned for the project, including the types of tests required, test documents, test methods, and test data collection. Specify each test (from unit testing through acceptance testing) in terms of entrance and exit criteria and the expected level of involvement (from the project management team, test group, and other functional areas).

Develop unit and integration scenarios to exercise and validate all specified application requirements, functions, and objectives. Plan the system and the acceptance tests to finalize the validity of the requirements.

An integrated load test combines scripts from the application to place a normal user load on the enterprise server infrastructure simulating production conditions on an average day. The number and types of users are estimated by the application teams consistent with the expected sustained load of the system.

Use controlled data with each type of test as specified in the test plan. Prepare the test data to include:

- Verifying the values of the functional capabilities of the software test component
- Identifying limitations and possible deficiencies
- Exercising its capabilities
- Verifying the component performs its intended function.

The Test Plan needs to indicate if pilot testing or a phased implementation is required for the project. In the case of phased software releases, include the requirements for regression testing as new elements are introduced.

Compare actual and expected results for each test. Identify discrepancies along with problem resolution. Retesting is required to verify the solution eliminates the problem and has not introduced new errors. Include a completed test report with the final test results. In addition, include the Error Log with signoffs by the testing personnel.

Outputs

Maintain the updated test plan in the project folder.

Review Process

The project team reviews any work performed by an independent testing group.

4.8.2 Identify Test Phases

Responsibility

Project Management Team

Applicability

Regardless of the testing techniques in use, the project will usually incorporate four test phases, as indicated below. Document any deviations from this basic strategy in the Test Plan.

Description

The software is tested in four sequential phases: unit, integration, system, and acceptance. Some projects may require additional types of tests. For example, prototype testing is required in some special situations. The four standard test phases, along with prototype testing, are described below:

- **Unit-Module Test Phase**

  This phase is performed by the developer creating the module and involves testing the individual software "units" or groups of related units or modules. A unit is a component not subdivided into other components. It is a logically separable part of a computer program. Evaluate the subject module to determine how well it meets the functional and performance requirements. Timing, memory, accuracy, and logical results are to be considered. Input data required for validating program logic, syntax, and performance requirements is prepared.

- **Systems Integration Test Phase**

  The project management team may perform system testing, or an independent test group may perform the testing with support from the project management team. This phase tests the completely integrated hardware and software to verify the product meets specified requirements and operates successfully.

  Integration testing is performed by the development team and constitutes an orderly progression in which software (and possibly hardware) elements are combined and tested to evaluate their interaction. Integration testing verifies the groups of functionally related modules interface properly and perform according to the requirements. Testing may include static testing which is the examination of source code to ensure the processing logic meets the requirements of the design and the application satisfies any explicit functional requirements.

  Integration and System testing are necessary to validate proper initialization, decision branching, and all functions specified in the Requirements Definition Document (RDD). Test the required outputs and interfaces to ensure proper interaction with other specified systems, along with transaction response times, machine resource allocation, and utilization.

- **Load Test Phase**

  Load testing observes production hardware and software configuration performance under a predetermined set of test scenarios to confirm the system, as built and deployed, can maintain adequate throughput, satisfactory response, and timely operation under different stress and volume conditions.

The purpose of Load Testing is to determine whether, or at what point, extreme conditions are likely to cause system failure.

- **Pre Systems Acceptance Test (Pre-SAT)**

  The development team may perform Pre-SAT testing depending on the complexity of the system and inherent constraints in the systems integration environment to be able to validate all the component connectivity and interfaces. Secondly, Pre-SAT can be used to validate the configuration to perform user acceptance testing.

- **User Acceptance Test Phase**

  Users may perform acceptance testing with assistance from the development team. However, team members may perform the testing with assistance from a designated user. Acceptance testing focuses on determining whether the software and related documentation satisfies the acceptance criteria. Testing includes all intra-system interfaces, documentation, procedures, and controls. The project sponsor may accept or reject the product based on test results.

Inputs

Inputs include:

- The statement of test techniques to be applied
- The statement of goals and objectives supplied from the planning phase.
- Any supplied use case scenarios

Outputs

Outputs include:

- The updated project plan including the specified test phases
- The updated test plan

Review Process

The updated project plan and test plan are reviewed by the project management team, the project sponsor, and users as appropriate.

4.8.3 Identify Test Environment Requirements

Responsibility

Project Management Team

Applicability

Specify a test environment for any project demanding a testing environment not already existing or for an existing environment likely to become stressed or taxed.

Description

Test environment requirements identify what is needed to perform testing activities throughout the project lifecycle, including: human resources, hardware, software, physical space, and other environmental requirements. Testing is to be performed on equipment similar to the production system. In some instances, this information is not known until the Detailed System Design (DSD) Phase.

Considerations during investigation of test environment requirements include:

- Evaluating automated testing tools for the:
    - Generation of test scripts
    - Creation of result and error repositories
    - Consideration of each tool's benefits and costs
    - Use of simulators
- Determining local area network, wide area network, and metropolitan area network testing environments, as needed
- Determining test lab, data generation, and error correction support
- Identifying Beta test sites

The Project Plan must be updated by the project management team to indicate hardware, software, and times to setup the environment(s).

Inputs include:

- The Project Plan
- The statement of goals and objectives
- The statement of test techniques to be used.

Outputs include:

- A detailed plan specifying the test environment becomes part of the test plan.
- The updated Project Plan

Review Process

Outputs are reviewed by the project management team, procurement staff, the project sponsor, staff supports specialists, and the architecture review board as appropriate.

4.8.4 Develop User Acceptance Test Plan

Responsibility

Project Management Team and Project Sponsor/primary user

Applicability

Any project, where user, project sponsor or other stakeholder review is considered necessary for approval, will require a user acceptance test plan. Exceptions would be small projects where the user requirements are clearly understood and where an expanded prototyping methodology is used where constant end user involvement ensures and guarantees a deliverable product when system testing is complete. In this case a user acceptance test plan would be moot.

Description

The Acceptance Test describes the test activities planned for project acceptance. The plan specifies the tests necessary to validate the project requirements have been met and the deliverables are in accordance with existing standards. In addition, the fully developed plan ensures a systematic approach to acceptance testing is established and the testing is adequate to verify functionality.

The complete set of system requirements and acceptance criteria form the basis of acceptance testing. Testing may be affected by features of the installation site and development. Special arrangements may be necessary when the software cannot be completely installed and executed in a live environment.

Note: Multiple configurations may require distribution to several installation sites.

Inputs

Inputs include:

- The Project Plan
- The detailed plan specifying the test environment from task 4.8.3 Identify Test Environment Requirements
- The project sponsor's statement of goals and objectives.

Outputs

Acceptance testing must be thoroughly documented with requirements traceability and acceptance criteria as authorized by the project sponsor. The completed User Acceptance Test Plan located in the Test Plan addresses the following requirements:

- Identification of the human resources involved in the acceptance test process with a description of their responsibilities, including the roles and responsibilities of any external test teams.
- Traceability of test designs and cases to software requirements
- Objectives and constraints for each test

- Complete test cases and test procedures, including inputs and expected outputs for each test case
- Descriptions of problem reporting, analysis, and resolution
- Locations where testing and tester training will take place
- Acquisition of any testing tools
- Estimation of resources and costs to accomplish testing

Maintain the initial Acceptance Test Plan in the Project Folder. This copy is refined during later phases as the project progresses. Review the initial plan during the Software Integration and Testing Phase and deliver as a final document.

The project management team updates the project plan to include the user acceptance test plan.

Review Process

Conduct structured walkthroughs as appropriate to ensure the initial User Acceptance Test Plan adequately describes all testing activities, test schedules, test products, test responsibilities, testing methodology, and required resources.

Reference/Resource

Structured Walkthrough Process Guide

4. 9 Conduct Phase Review

4 9.1 Conduct Peer Reviews

Peer reviews are conducted within the project management team to review each of the Requirements Definition Phase deliverables. This is not a formal process but rather a review to ensure team members at the same level agree to the deliverables produced and to leverage the knowledge and experience of others.

4.9.2 Conduct Structured Walkthroughs

Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of outputs and deliverables. The walkthroughs are scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage determines the frequency of the walkthroughs.

Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants. In some cases, it may be beneficial to include software users in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same. For further information, refer to the Structured Walkthrough Process Guide.

4.9.3 Conduct In-Phase Assessments (IPA)

If an IPA is required for this project, the work will be performed by an independent reviewer. The reviewer delivers the IPA report to the project management team which maintains the document in the Project Folder. For further information, refer to the In-Phase Assessment Process Guide.

4.9 4 Conduct Phase Exit

A Phase Exit is conducted at the end of each phase of development or maintenance. The process ends with the receipt of concurrence from the designated approvers to proceed to the next phase. Concurrence indicates all known issues have an acceptable plan for resolution. The project management team maintains the document in the Project Folder. For further information, refer to the Phase Exit Process Guide.

5.0 General System Design (GSD)

Applicability

The System Development Methodology (SDM) allows the use of different design and construction techniques. 4.2 Select Requirements Analysis Technique addresses the issue of matching the analysis technique with the project at hand. Similarly, the software design technique is based upon the unique aspects of the project, along with project and organizational constraints.

2.0 Software and Technology Engineering Process provides examples of how various projects may be designed. The methodology allows for combining and eliminating tasks when appropriate. For example, a rapid prototyping project may combine the Requirements Definition and General System Design (GSD) phases, resulting in a combined phase. The combination of phases may be performed multiple times as part of an iterative development effort.

Today's enterprise development organizations are seeking to reuse analysis, design, code, and executables, in compliance with the Information Resource Management (IRM) Strategic Plan for modularity, interoperability, and reusability**.**

Software developed with Visual C++, Java, Visual Basic, or ActiveX components benefit from newer analysis and design techniques. The methodology encourages the use of Object-Oriented (O.O.) and object-based methods, but does not dictate which, if any, to use. This decision must be based on many factors relating to the project at hand. Various factors may affect the selection of analysis and design techniques, just as the techniques affect the project tasks and deliverables.

The tasks may be modified or omitted as appropriate. Deviations are permitted, provided customer requirements are satisfied and have received management approval.

Description

The General System Design document maps the *"what to do"* of the Requirements Definition Document (RDD) to the *"how to do it"* of the design specifications. During this phase, the overall structure of the software is defined from a functional viewpoint. The GSD describes the logical system flow, data organization, system inputs and outputs, processing rules, and operational characteristics of the software from the user's point of view. It is not concerned with the software or hardware supporting the operation. Neither is it concerned with the physical organization of the data or the programs that accept the input data, execute the processing rules, and produce the required output.

The focus in this phase is on the functions and structure of the software. The project sponsor and users must be able to understand the design. One objective is to define and document the functions of the software to the extent necessary to obtain approval. The GSD must also include enough detail for the Detailed System Design (DSD) to be developed.

Prototyping the system functions is helpful in communicating the design to the project sponsor and users. By demonstrating the appearance of screens and limited navigation and functionality the project sponsors are often able to grasp exactly what it is the IT staff has in mind based upon the requirements. This method can often be invaluable in clearing up and identifying miscommunications early in the project. Prototypes are used to simulate one function, a module, or the entire software, and are useful in the transition from the GSD to the DSD.

Inputs

Inputs include the:

- Work Plan
- Requirements Traceability Matrix
- Requirements Definition Document (RDD)
- Description of Analysis Technique
- Test Plan

High-Level Activities

This section is divided into sub-sections describing the high-level activities performed during this phase. The activities represent the minimum requirements for a large software development project. Notes are provided to assist in customizing the lifecycle phase requirements to accommodate project scalability. The high-level activities for the GSD Phase consist of:

5.1 Select Design Technique

5.2 Determine Software Structure

5.3 Develop General System Design (GSD)

5.4 Design the User Interface

5.5 Create Data Model

5.6 Create Logical Data Model

5.7 Manage Requirements

5.8 Design System Architecture

5.9 Develop Capacity Plan

5.10 Initiate Procurement of Hardware and Software

5.11 Develop Training Plan

5.12 Develop Conversion Plan

5.13 Develop Architecture Review Board (ARB) Document

5.14 Conduct Phase Review

Outputs

The content and packaging of the outputs may vary depending on project size and complexity. Typical requirements for a large software project include:

- Description of Design Technique
- General System Design (GSD) Document
- Requirements Traceability Matrix (*expanded*)
- System Architecture Document
- Capacity Plan (*initial*)
- Acquisition and Installation Plan (initial)
- Training Plan (*initial*)
- Conversion Plan (*initial*)
- Architecture Review Board Document
- Test Plan (revised)

A matrix showing the outputs associated with each high-level activity is provided in Exhibit 5.0-1, General System Design High Level Activities and Outputs.

Review Process

Quality reviews are necessary during this phase to validate the various outputs. The activities appropriate for quality reviews are identified in this phase and 2.3 Quality Reviews. This review is an important milestone in the design process. The time and resources needed to conduct the Architecture Review Board (ARB) and GSD Review are reflected in the project resources, schedule, and work plan.

Reference/Resource

2.3 Quality Reviews, provides an overview of the types of reviews to be conducted within a project.

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

Checklist

General System Design Phase Checklist

Exhibit 5.0-1 General System Design High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
5.1	Select Design Technique	Description of Design Technique	N
5.2	Determine Software Structure	Design Entities and Dependencies	Y
5.3	Develop General System Design	General System Design Document	Y
5.4	Design the User Interface	General System Design Document (User Interface)	Y
5.5	Create Data Model	General System Design Document (Data Dictionary)	
5.6	Create Logical Data Model	General System Design Document (Logical Data Model)	Y
5.7	Manage Requirements	Requirements Traceability Matrix (*expanded*)	Y
5.8	Design System Architecture	System Architecture Document (*initial*)	Y
5.9	Develop Capacity Plan	Capacity Plan (*initial*)	Y
5.10	Initiate Procurement of Hardware and Software	Acquisition & Installation Plans	N
5.11	Develop Training Plan	Training Plan (*initial*)	Y
5.12	Develop Conversion Plan	Conversion Plan (*initial*)	Y
5.13	Develop Architecture Review Board Document	Architecture Review Board Document	Y
5.14	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

5.1 Select Design Technique

Responsibility

Project Management Team

Applicability

The project management team selects a design technique to coordinate with the Department of Human Services (DHS) architectural and design standards. At times, the selection may be influenced by funding stipulations. This work can often be done during the requirements phase when the experience and skills exist within the team members involved, but it is logically considered part of the general design phase.

Description

A systematic approach to building the General System Design (GSD) and Detailed System Design (DSD) simplifies the process and fosters a reliable, maintainable, and testable system. A complete design technique includes:

- A description of the method or methods used in creating and refining the design
- Compatibility with the requirements analysis technique and the automated tools used by the project management team
- Straightforward rules relating information obtained during requirements analysis to a distinct software structure
- Design standards complying with the site's current software development practices, the project sponsor organization's standards, and the constraints imposed by the software tools and hardware platform
- A practical approach to design is amenable to a wide variety of software
- The development of small, intermediate design products can be used to measure quality and progress
- Tools to be used in the design process
- Well-defined measures to assess the quality of the design
- Guidance on how to create a design to minimize the required maintenance and to maximize the usability of the applications components.

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

Sample Design Methods

Examples of some common design techniques are:

- Function-oriented design methods model the applications software by dividing into components, identifying the inputs required by those components, and identifying the outputs produced. Function-oriented design methods include structured analysis and structured design. The major models or design representations used by this method are data flow diagrams, data dictionaries, structure charts, and process specifications.
- Data-oriented design methods use program structures derived from the data structures. Tree diagrams are typically used to represent both the data and the program structures.
- Object-oriented design methods produce a software architecture based on the objects manipulated by systems or subsystems rather than by functions. An Object Oriented (O.O.) design closely resembles a model of reality since it presents an abstraction of the real-world objects and the operations performed on or by them. The design structure tends to be layers of abstraction where each layer represents a collection of objects with limited visibility to other adjacent layers.

Inputs

- The Requirements Definition Document (RDD).
- The Project Plan
- The tools used in the development of the requirements

Outputs

Maintain a description of the design technique in the Project Folder and distribute to the project management team, project sponsor, and involved users.

Review Process

Conduct a structured walkthrough to verify that the design technique is appropriate for the scope and objectives of the project. A structured walkthrough is not required when the choice is obvious or when it has been used successfully on similar projects.

Reference/Resource

Structured Walkthrough Process Guide

5.2 Determine Software Structure

Responsibility

Project Management Team

Applicability

Although some newer software development techniques [e.g., Object-Oriented (O.O.)] discourage the use of top-down design, the approach is still used by various structured methods. Project management may find team participants are more comfortable with a top-down approach to software structuring. As in any project, human, as well as technical factors must be considered.

Although there are good arguments for developing systems using one of the newer techniques, the developers' experience and abilities must be considered. Management may find developers with new skill sets have already used, or are able to use, newer design techniques.

> *Note: Technical considerations are important. Some smaller projects may be better served with a top-down approach. Ensure the selected development techniques are congruent with the development tools.*

Description

Determining the Software Structure is important at the design stage of development to eliminate unforeseen costs. The software methodology can be structured as being comprised of the software design process component and the software design representation or diagrammatic component. The process component is based on the basic principles established in the methodology while the representation component is the blueprint from which the code for the software will be built.

> *Note: In practice, the design methodology is often constrained by existing hardware configuration, the implementation language, the existing file and data structures and the existing organization practices, all of which would limit the solution space available to develop the software.*

The design team needs to meticulously record or diagram the software design including the basis for the choices made. This is important for the purposes of walkthroughs and maintenance.

Various methods exist to determine the software structure. Some of these methods are Top Down/Bottom-Up Design, Stepwise Refinement Design, Structured Design, and Object-Oriented Design. However, the two most distinguish approaches are broadly identified as the Hierarchical Design and Object-Oriented Design.

A. Hierarchical Design

A hierarchical approach is useful for determining the structure and components of the software. Software system decomposition is a type of hierarchical approach dividing the software system into different levels of abstraction. Decomposition, an iterative process, continues until single purpose components (i.e., design entities or objects) can be identified. Decomposition is useful for understanding the structure of the software along with the purpose and function of each entity.

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

There are several reliable methods for performing system decomposition. Select a method to enable the design of simple, independent entities.

Identify Design Entities

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

A design entity *attribute* is a characteristic or property of a design entity. It provides a statement of fact about an entity. The list below may prove helpful in assigning common attributes, but is not meant to supersede any part of a packaged technique or design methodology:

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

Identify Design Dependencies

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

Identify the dependent entities of the software Detailed System Design (DSD), describe the coupling, and identify the resources required for the entities to perform the function. In addition, define the strategies for interactions among design entities and provide the information needed to perceive *how*, *why*, *where*, and *at what level* actions occur.

Dependency descriptions provide an overall picture of how the software will work. Data flow diagrams, structure charts, and transaction diagrams are useful for showing the relationships among design entities. The dependency descriptions may be useful in producing the System Integration Plan by identifying entities along with the required order for the development. Dependency descriptions can also be used to aid in the production of integration test cases.

B. Object-Oriented (O.O.) Design

For O.O. and object-based designs, the General System Design (GSD) centers on the idea objects can be designed to exist on a virtual machine. Limitations based on computing technology and the determinations of the location of objects (i.e., client or server) are not considered at this point.

Algorithmic properties of classes, based on the O.O. analysis, are presented in the GSD.

Some important issues to be considered if an O.O. or object-based design method is to be employed include:

- What technique was used to perform the analysis?

  If a functional approach was used in the analysis phase, then functional limitations such as extendibility, explicit reference, reusability, and maintenance difficulty shall be taken into consideration to pursue an object-oriented O.O. design and implementation.

- Which design methods are being considered?

  Not all O.O. methods are the same. Depending on the method used, some of the functional and physical design tasks may be combined.

- Are short iterative cycles being incorporated in the lifecycle?

  Rapid design/build cycles may not fit well with a separately developed functional model for each iteration.

  Note: Project tasks, deliverables, and design methods must be aligned and consistent.

Outputs

Record the design entities, integrate them into the GSD Document, and maintain the document in the Project Folder.

Review Process

Conduct structured walkthroughs, as needed, to verify the design entities and dependencies are correct, complete, and possess the required attributes.

Reference/Resource

Structured Walkthrough Process Guide

5.3 Develop General System Design (GSD)

Responsibility

Project Management Team

Applicability

The design is used for communicating software design information to facilitate planning, designing, and constructing decisions. The GSD is widely used for quality assurance purposes.

Description

The GSD is a model describing how the software will be structured to satisfy the requirements identified in the Requirements Definition Document (RDD). It presents a partitioning of the software system into design entities and describes the important properties and relationships between the entities.

Organize and present the entities within the GSD in a way the project sponsor and users understand them. The goal is to compile the entities and the attributes in a manner facilitating the access of the design information from various viewpoints (e.g., project management, configuration management, quality assurance, and testing).

Outputs

Each requirement identified in the GSD Document must be traceable to one or more design entities. This traceability ensures the software will satisfy the requirements and will not include inappropriate or extraneous functionality. The Requirements Traceability Matrix developed in the Requirements Definition Phase is expanded to relate the GSD to the requirements. The expanded matrix is maintained in the Project Folder.

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix to ensure all required functionality is included.

Reference/Resource

Structured Walkthrough Process Guide

Expanded Requirements Traceability Matrix developed in the Requirements Definition Phase.

Tasks

The tasks for developing the GSD consist of:

5.3.1 Develop General System Design (GSD) Document

5.3.2 Conduct General System Design (GSD) Review

5.3.1 Develop General System Design (GSD) Document

Responsibility

Project Management Team

Applicability

The purpose of this document is to explain ‘how’ the requirements will be satisfied by the new system.

Description

The GSD Document is developed describing system functions in user terms. Although, there is a great amount of flexibility, the GSD approach must be consistent with the analysis approach selected in 4.2 Select Requirements Analysis Technique. The document is written from the system users’ perspective and provides the project sponsor and users with an opportunity to review and offer input in this early design phase.

Outputs

The GSD Document is prepared using the organizational tools available. The format may vary somewhat depending upon the analysis and design techniques employed but should stay fairly close to the document template found in the Department of Human Services (DHS) Standards Business Domain. Once approved, the document serves as authorization to use the GSD as a basis for the Detailed System Design (DSD).

Review Process

Conduct a structured assessment to ensure the GSD Document is accurate, complete, and describes the GSD in a manner understood by the project sponsor and users.

Submit the draft document to the project sponsor and users for review and approval. After making any necessary changes, the approved GSD Document becomes an official agreement and basis for DSD.

The completion of the GSD Document is an appropriate time to schedule an In-Phase Assessment (IPA).

Template

General System Design (GSD) Document

Reference/Resource

In-Phase Assessment Process Guide

5.3.2 Conduct General System Design (GSD) Review

Responsibility

Project Management Team

Applicability

A review of the GSD is needed following an existing project, new project, or COTS-based project. This step should always be applicable whenever a GSD is created.

Description

The GSD Review is a formal technical review of the basic design approach. The primary goal of the review is to demonstrate the ability of the software design to satisfy the project requirements. The review typically consists of a series of presentations by the project management team to the project sponsor, users, functional area points-of-contact, and quality assurance representative. Vendors may be invited to participate when a Commercial Off the Shelf (COTS) product is being considered for the system architecture.

The GSD Review verification process includes:

- Evaluate the progress, technical adequacy, and risk resolution of the selected design approach. Determine whether the project management team is following the approved design approach.
- Establish the existence and compatibility of the physical and functional interfaces.
- Determine whether the GSD embodies all of the requirements.
- Verify the design represents software meeting the functional, data, and interface requirements.
- Review the planned user interfaces to the software.
- Identify potential high-risk areas in the design and any requirement changes to reduce risk.
- Consideration has been given to optimizing the maintainability and maintenance aspects.

Review Items

Items to be considered for review and evaluation during the GSD review include:

- **Functional flows** – Indicate how the computer software functional flows map the software and interface requirements to the individual high-level components of the software product.
- **Storage allocation data** – Describe how available storage is allocated to individual software components. Timing, sequencing requirements, and relevant equipment constraints are used in determining if the allocation needs to be included.
- **Control functions** – Describe the executive control and start/recovery features of the software.

- **Security** – Identify the security requirements and provide a description of the techniques used for implementing and maintaining security within the software product.
- **Software development facilities** – Describe the availability, adequacy, and planned utilization of the software development facilities including both Government-provided and commercially available facilities.
- **Software development facility vs. the operational system** – Describe any unique design features existing in the GSD in order to allow use within the software development facility not existing in the operational software. Provide the information generated on the design of support programs not explicitly required for the operational system to assist in the development of the software.
- **Development tools** – Describe any special tools (e.g., simulation, data reduction, or utility tools) planned for use during software development. (Do not include deliverables.)
- **Commercial resources** – Describe commercially available computer resources, including any optional capabilities (e.g., special features, interface units, special instructions, controls, formats). Identify any limitations of commercially available equipment (e.g., failure to meet user interface, safety, and maintainability requirements) and identify any deficiencies.
- **Existing documentation** – Maintain a file and have available for review any existing documentation supporting the use of commercially available computer resources.
- **Support resources** – Describe the resources necessary to support the software during engineering, installation, and operational state (e.g., operational and support hardware and software personnel, special skills, human factors, configuration management, testing support, documentation, and facilities/space management).
- **Standards** – Describe any standards or guidelines to be followed.

Outputs

A file note is produced from each meeting and distributed to the attendees and project stakeholders. The notes consist of significant questions and answers, action items, individual/group responsibilities, deviations, conclusions, and recommended courses of action resulting from presentations or discussions.

Recommendations not accepted need to be recorded along with the reason for non-acceptance. Distribute the file notes to the project sponsor and users. The options for the review performance are:

- *Approval* – indicates the GSD is satisfactorily completed.
- *Contingent Approval* – indicates the GSD is not considered complete until the satisfactory completion of resultant action items.
- *Disapproval* – indicates the GSD is inadequate. Another GSD Review is required.

Review Process

Checklist of Items to be Reviewed

Description	Yes	No	Comments
Functional Flows			
Storage Allocated Data			
Control Flows			
Securities			
Software Development			
Software Development Facilities			
Software Development Facilities vs. Operational Systems			
Development Tools			
Commercial Resources			
Existing Documentation			
Support Resources			
Standards			
Approval / Contingent / Disapproval Signature			

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

5.4 Design the User Interface

Responsibility

Project Management Team

Applicability

This section presents steps to take during user interface design along with generally accepted design principles. The sub-sections are used in conjunction with any relevant standards, or as a substitute if none exists. If no user interface exists or is involved in this project, this step can be skipped.

Description

Design the user interface appropriate for the users, content, and operating environment of software. Include a specification of interface levels for all categories of users. For interactive user environments, user interface prototypes may be developed and reviewed with user representatives. Prototypes have been successfully used to uncover problems early in the development process and to gain user acceptance.

A common problem from a user’s perception is a prototype can become a working product with “just a few fixes.” Take extra care to manage expectations by explaining the use of prototypes and the unfitness for production environment.

Review the standard every time new software is planned to verify the user interface is compatible with the selected system architecture. For example, a Microsoft Windows® user interface standard is not appropriate for a web-based application.

Outputs

The user interface design documentation which includes the Interface Flow Diagram, if created.

Review Process

Project Management Team reviews the User Interface Design and ensures it is in compliance with DPW standards.

Reference/Resource

Web Application Development Guide

5.5 Create Data Model

Responsibility

Project Management Team and a Database Analyst

Applicability

A member of the Database Design section, working as a project management team member, performs this activity. The procedures for the section are to be followed in coordination with system development. The information is provided as a reference and is not meant to supersede any established procedures in the department. This step should always be performed when data elements are going to be added, modified or deleted in any significant way. This is necessary to ensure adherence to good design practices and to any database properly scalable.

Description

A data model is a representation of a collection of data objects and the relationships between the objects. The functions of the data model are to:

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

Outputs

The data dictionary first developed in the Requirements Definition Phase is now expanded to catalog every known data element to be used. Detailed information about the data elements include attributes, known constraints, input sources, output destinations, and known formats.

Although it might prove too technical for some users, there are cases when the data dictionary serves as a central repository of information for both application developers and end users. The dictionary includes business rules, processing statistics, and cross-referencing information for multiple vendor environments.

To expand the data dictionary, the steps used to define, analyze, and complete the data definitions include:

- Identify data needs associated with various system features
- Match (verify) data needs with the data dictionary
- Match the data dictionary with specific data structures
- Create data record layouts
- Ensure all data can be maintained through add, change, or delete functions

The data dictionary will be further refined in the Detailed System Design (DSD) Phase to complete the information on data elements, entities, files, physical characteristics, and data conversion requirements.

Sample Attributes

The attributes (information) to include for data dictionary elements are:

- Add/Change/Delete
- Table Name
- Description
- English Name
- Business Rules
- Is Application a Reference Table

Review Process

Conduct a structured walkthrough to verify the data dictionary is correct and complete. Validate the data model against any department or site specific data models.

Reference/Resource

Structured Walkthrough Process Guide

Examples of Schema Diagrams

Instructions for Completing the Data Dictionary Excel Spreadsheet

5.6 Create Logical Data Model

Responsibility

Project Management Team

Applicability

This activity is typically performed by projects incorporating structured (i.e., functional decomposition) methods where data are normalized into a logical data model.

Description

The logical data model defines the flow of data through the software system and determines a logically consistent structure for the software. Each module defining a function is identified, interfaces between modules are established, and design constraints and limitations are described. The logical data model focuses on the real world problem to be solved.

A logical data model is used to explore the domain concepts and the relationships. Logical Data Models depict the logical entity types, typically referred to as entity types, the attributes describing the entities, and the relationships between the entities.

When creating a logical data model, the organization of the model follows the natural organization of the software subject matter. The names given to the components of the model must be specific. The connections between the components of the model need to be as simple as possible.

Outputs

The logical data model is documented in user terminology and contains sufficient detail to obtain the project sponsor's and users' approval. Use data flow diagrams to show the level of detail needed to reach a clear, complete picture of processes, data flow, and data stores.

Maintain up-to-date logical data model and data flow diagrams in the Project Folder and incorporate into the General System Design (GSD) Document. The logical data model and data flow diagrams will serve as a resource for planning enhancements during the Maintenance Phase.

Review Process

Conduct a structured walkthrough to verify the logical data model is correct, logical, and complete.

Reference/Resource

The use of Use Cases Diagrams or any Unified Modeling Language (UML) is practical in this area to determine the Logical Data Model and the Data Dictionary.

Structured Walkthrough Process Guide

ITP-INF003 Data Modeling Standards

5.7 Manage Requirements

Responsibility

Project Management Team

Applicability

At the completion of the General System Design (GSD) Document, review and cross-reference all the requirements in the GSD and the Requirements Traceability Matrix.

Description

The process of eliciting, documenting, organizing, and tracking changing requirements in this phase is communicated to the project management team to ensure the iterative and unanticipated changes are maintained throughout the project lifecycle. The focus of managing requirements is to aid the stakeholders and maintain a shared vision. This includes:

- Understanding the relationships among key stakeholders and involving them
- Identifying requirements change
- Managing the changes to requirements
- Identifying and tracking requirements attributes
- Tracing requirements

Outputs

Each requirement identified in the Requirements Definition Document (RDD) must be traceable to one or more design entities. This traceability ensures the software satisfies the requirements and does not include inappropriate or extraneous functionality. The Requirements Traceability Matrix developed in the Requirements Definition Phase is expanded to relate the GSD to the requirements. The expanded matrix is maintained in the Project Folder.

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix to ensure all required functionality is included.

Reference/Resource

Structured Walkthrough Process Guide

- Elicitation Techniques such as Interviews, Scenarios, Prototypes, Facilitated meetings or observations
- Support tools such as spreadsheets and simple databases developed in house could be used to support the Manage Requirements activity

5.8 Design System Architecture

Responsibility

Project Management Team

Applicability

Designing system architecture is important in an effort to produce an efficient and cost effective application.

Description

The system architecture encompasses the technical and application architecture of the application. The application architecture provides the design and layout of the requirements specific to the system application as a whole (coding standards, application framework usage). The technical architecture provides the technical requirements (hardware, software, server configuration) for the system application to work.

Outputs

The system architecture document is designed and documented in accordance with the project design standards. Submit the designs to the project sponsor, users, and possibly the system administrator for review and approval. Incorporate the approved design into the General System Design (GSD) Document and maintain the document in the Project Folder.

Review Process

Conduct a structured walkthrough, as needed, to verify the System Architecture designs are correct and complete.

Reference/Resource

Structured Walkthrough Process Guide

Architecture Review Board

Tasks

The tasks for designing the system architecture consist of:

5.8.1 Design System Interfaces

5.8.2 Design System Security Controls

5.8.3 Evaluate System Architecture Alternatives

5.8.4 Recommend System Architecture

5.8.1 Design System Interfaces

Responsibility

Project Management Team

Applicability

Design of Systems Interface is crucial to promote ease of use of the systems as well as increasing flexibility for changes and improvements of the systems. Some of the objectives of designing the systems interfaces earlier in the General System Design (GSD) phase are: to reduce the frequency of error handling and redesign in process flow.

Description

The system interfaces depict how the software will interface with other systems. The interfaces are based on the system interface requirements documented in the Requirements Definition Document (RDD).

Sample Issues

Issues to consider when designing the system interfaces include:

- System inputs and outputs
- Method of interface
- Volume and frequency of data
- Platform of interfacing system
- Format of data
- Automatic or manual initiation of interface
- Verification of data exchange
- Validation of data

Outputs

The Detailed System Design (DSD) interfaces are designed and documented in accordance with the project design standards. Submit the designs to the project sponsor, users, and possibly the system administrator for review and approval, for each system interfacing with the software.

Any incompatibilities with interfaces need to be identified early in the design process so corrective actions can be taken. Incorporate the approved design into the General System Design (GSD) Document and maintain the document in the Project Folder.

Review Process

Conduct a structured walkthrough, as needed, to verify the system interface designs are correct and complete.

Reference/Resource

Structured Walkthrough Process Guide

5.8.2 Design System Security Controls

Responsibility

Project Management Team and Security Personnel

Applicability

This task is typically performed by projects when the system security is needed.

Description

The security and access requirements documented during the Requirements Definition phase are used to design the security controls for the software. Security controls must be designed in conjunction with any relevant security standards, as specified by the Governor's Office of Administration/Office for Information Technology (OA/OIT) Security Policies and the Department.

Outputs

System security controls are designed and documented in accordance with project design standards. Present the controls to the project sponsor and users for review and approval. Include the approved design in the General System Design (GSD) Document and maintain the document in the Project Folder.

Review Process

Conduct a structured walkthrough to verify the system security controls are correct and complete. A representative from the security group is required to attend the walkthrough.

Reference/Resource

Structured Walkthrough Process Guide

The Data Exchange Guidelines are located under the IRM Standards, Business and Technical Standards, Integration and Middleware Domain.

5.8.3 Evaluate System Architecture Alternatives

Responsibility

Project Management Team

Applicability

System Architecture Alternatives are needed to balance the budgeting obligation; to accommodate the size of the project scope and resources; and to ensure platform compatibility.

Description

An evaluation of architectural alternatives must consider many factors, for example, the experience the project management team members have with each alternative and the availability of reusable components to facilitate the implementation. Issues to be considered when evaluating the alternatives include:

- Identification of which functions (or portions of functions) are to be automated (include an examination of *what* the portion(s) will encompass) and which will be manual.
- The technical solution for the objectives. The determinations of *how* the software is to be designed (e.g., online vs. batch, client-server vs. mainframe, Oracle vs. Sybase).
- The owner and users computing environment and needs created by the technical solution. Consider any hardware and software to be acquired, including system software and database management products.

One approach for evaluating the architecture alternatives (reasonable, alternative methods may be used) is to:

- Conduct a Cost Benefit Analysis to determine the most cost-effective alternative. On the benefits side, include the improvements over the current process used to support the business application. On the cost side, include any impact from current capabilities along with the rationale for allowing the impact.
- Generate and evaluate a data flow diagram for each alternative.
- Identify how users interact with the features associated with each alternative (such as the generation of queries and reports).
- List the risks associated with each alternative and develop a plan for mitigating each risk.
- Compare the performance capabilities of each alternative. For example, *"how fast will each alternative be able to process the users' work given a particular hardware resource?"* Performance is usually expressed in terms of throughput, run time, or response time. Five factors frequently affecting performance include:
  - Number of intermediate files in a system
  - Number of times a given file is passed
  - Number of seeks against a disk file

    - Time spent in calling programs and other system overhead
    - Time taken to execute actual program
- Compare the security and access control features of each alternative, including the extent to which each alternative provides protection against human errors, machine malfunction, or deliberate mischief. Common controls include:
    - Check digits on predetermined numbers
    - Batch control totals
    - Creation of journals and audit trails
    - Limited access to files
- Compare the ease which each alternative allows the system to be modified to meet changing requirements, such as:
    - Fixing errors
    - Changing user needs
    - Mandatory/statutory modifications
    - Enhancements

Outputs

Records for each alternative evaluated are maintained in the Project Folder and used to develop a summary of the system architecture alternatives. The summary will be integrated into the materials presented to the project sponsor when a system architecture recommendation is made.

If a Cost Benefit Analysis is conducted, prepare a report describing the process used for the analysis, a summary of the alternatives considered, the results obtained, and maintain the document in the Project Folder. The report will be integrated into the materials presented to the project sponsor when a system architecture recommendation is made.

Review Process

Structured walkthroughs are conducted to review the records of each alternative.

Reference/Resource

Structured Walkthrough Process Guide

5.8.4 Recommend System Architecture

Responsibility

Project Management Team

Applicability

Based upon the conclusion of the architecture alternatives process, the recommended system architecture is rendered.

Description

Based on the results of the architecture alternatives evaluation, a recommendation is developed for a cost-effective system architecture satisfying the project requirements. Prepare a presentation for the project sponsor and users providing supporting information to:

- Review the limitations or problems with any current manual or automated process resolved by the software.
- Present the logical data model for the software. Highlight new functionality to be implemented.
- List each architecture alternative evaluated, including:
  - A description of the alternative
  - An overall data flow diagram showing how the alternative is implemented
  - The way the system looks to the users, in terms of hardware, user interface, reports, and query facilities
  - The estimated benefits of the alternative
  - The estimated cost and time to implement the alternative
  - A statement of the element of risk associated with the alternative
- Present the recommended alternative and explain why it was selected.

Before proceeding to the next phase of development, concurrence must be reached regarding the selection of the system architecture. This is done by either formally accepting the recommendation or by directing the team to use a different architecture. Any delay in making this decision results in a lengthening of the project schedule.

Outputs

The recommended system architecture and a summary of each alternative are documented in the System Architecture Document. The recommendation includes any background information relevant to the decision process, such as a Cost Benefit Analysis Report. The rationale for proposing the recommended architecture is described along with an analysis of the impact on the users' organization. The impact analysis includes existing and planned systems.

The architecture recommendation is presented to the project sponsor and users, either as a presentation or a distributed document. The document is maintained in the Project Folder.

Review Process

A structured walkthrough is conducted to review the completeness and accuracy of the evaluations and recommendation.

Template

System Architecture Document

Reference/Resource

Structured Walkthrough Process Guide

5.13 Develop Architecture Review Board Document

5.9 Develop Capacity Plan

Responsibility

Project Management Team

Applicability

The Capacity Plan is used to identify the capacity constraint of the project.

Description

The Capacity Plan includes estimates for the application usage, network bandwidth, disk storage capacity, demographic profile environment, load test environment, production environment, and batch/File Transfer Protocol (FTP) capacity. The plan is continually revised throughout the project to ensure the latest capacity requirements are captured and verified.

Outputs

The capacity plan is designed and documented in accordance with project design standards. Maintain the Capacity Plan in the Project Folder.

Review Process

Conduct a structured walkthrough to verify the capacity is correct and complete.

Template

Capacity Plan

Reference/Resource

Structured Walkthrough Process Guide

5.10 Initiate Procurement of Hardware and Software

Responsibility

Project Management Team

Applicability

This activity is completed when information is needed for the requisition on organizational hardware and software requirements.

Description

The procurement of any hardware or software must be initiated well in advance of the planned need for the products. Of course, this can't begin until enough information is known about the ultimate design of the product. However, adequate time must be allotted in the Project Plan timeline for the selection, procurement, installation, testing, and training associated with each vendor product.

Careful consideration is given to purchasing off-the-shelf software before expending the time, resources, and costs associated with developing custom-built systems. Whenever possible, acquire off-the-shelf software to satisfy some or all of the project requirements. In addition, some projects require the acquisition of hardware or software to support designing, coding, and testing activities.

Attempt to acquire a demonstration package of any proprietary software before completing the design specifications. The software may prove inadequate or inappropriate once it has been evaluated through hands-on use. Develop a pilot of the software to exercise the most important functions provided by the proprietary software. This will provide definite performance indications.

When the expected operating platform for the product requires extensive procurement of hardware and software, it is recommended the procurement needs be addressed as early in the lifecycle as possible. If hardware and software acquisition requirements are known, the Acquisition and Installation Plans may be developed for all operating sites. The Acquisition and Installation Plans are reviewed, and revised if necessary, at the beginning of the Development Phase.

> *Note: Requirements for the Acquisition and Installation Plans are provided in the, Development Phase.*

For additional information on developing an acquisition plan, refer to 7.1 Develop Acquisition Plan.

Outputs

Acquisition and Installation Plans, along with all software and hardware procurement records, are maintained in the Project Folder.

Review Process

Not required; however, a peer review of software and hardware procurement records can be beneficial to ensure the correct order is placed.

5.11 Develop Training Plan

Responsibility

User Education Team and Project Management Team

Applicability

A Training Plan is needed for post implementation education.

Description

A Training Plan defines the training needed to successfully implement, operate the software, and address the training provided to the project sponsor, users, and maintenance staff.

Training must address both the knowledge and the skills required to use the system effectively. The objectives of the training program are to:

- Provide trainees with the specific knowledge and skills necessary to perform the work.
- Prepare training materials to “sell” the software as well as instruct the trainees. The training program’s goal is ensure the trainees become familiar with the new software.
- Account for the knowledge and skills the trainees possess and use this information as a transition to learning new material.
- Anticipate the needs for follow-on training after the software is fully operational, including the possibility of refresher courses, advanced training, and repeats of basic courses for new personnel.
- Build in the capability to easily update the training as the software evolves.
- Address the need for extra training if new hardware or software is introduced.

Include the project sponsor and user representatives in the planning activities to determine the training needs for all categories of users (managers, users, and maintenance staff).

Outputs

At a minimum, the initial Training Plan addresses:

- Identification of personnel to be trained. Verify the list with the project sponsor and users.
- A definition of the overall approach to training along with required courses.
- Delineation of the scope of the training needed for users, management, operations, and maintenance personnel.
- Explanation of how and when training will be conducted, including instructor qualifications, learning objectives, and mastery or certification requirements (if applicable).
- Identification of any skill areas for which certification is necessary or desirable.
- Establishment of a preliminary schedule for the training courses. The schedule must reflect training requirements and constraints outside of the project. It may also include the identification of critical paths in the training schedule such as the period for the software’s installation and conversion to production status.

- Definition of the required courses, including an outline of content and sequence.

In addition, it may include:

- Reference to the organization’s training policy for meeting training needs
- Steps to ensure software managers have received orientation on the training program
- Ensure training courses prepared at the organization level are developed and maintained according to organizational standards
- A waiver procedure for required training for individuals who already possess the knowledge and skills required to perform their designated role
- Measurements to be used for evaluating the effectiveness of the training
- Ensure training records are properly maintained

The initial Training Plan is maintained in the Project Folder. The plan will be reviewed and updated during the Software Integration and Testing Phase.

Review Process

Conduct a structured walkthrough to ensure the initial Training Plan is accurate and complete.

Template

Training Plan

Reference/Resource

Structured Walkthrough Process Guide

5.12 Develop Conversion Plan

Responsibility

Project Management Team

Applicability

The Conversion Plan is created when there is a need for software replacement.

Description

If the software replaces an existing automated system, a Conversion Plan is generated. The major elements of this plan are conversion procedures, outline of the installation of new and converted data stores, conversion programming and implementation planning.

File conversion includes a confirmation of file integrity, as well as a determination of the output of the new system compared with the current system. The objective of file conversion is for new files to be complete, accurate, and ready to use.

Many factors influence data conversion, including the design of the current and new system, and processes for data input, storage, and output. Understanding the data's function in the current system and the new system is very important. The structure of the data to be converted can limit the development of the system and affect the choice of software.

Outputs

The Conversion Plan identifies what conversions are needed and how the conversion(s) will be implemented. Factors to be considered include:

- Portions of the conversion process may be performed manually.
- Parallel runs of the old and new systems may be necessary during the conversion process.
- The function of the data in the current and new systems must be understood. This will not always be the same.
- The order the data is processed in the two systems may influence the conversion process.
- Volume considerations, such as the size of the database and the amount of data to be converted, influence how the data will be converted.
- User work and delivery schedules, timeframes for reports and end-of-year procedures, and the criticality of the data help determine when data conversion is scheduled.
- A determination of data availability during the conversion process to determine data access limitation.
- Data purification tasks are performed.
- Plan for the disposition of obsolete or unused data not converted.

Review Process

Structured walkthroughs are conducted, as needed, to ensure the Conversion Plan is accurate and complete.

Template

Conversion Plan

Reference/Resource

Structured Walkthrough Process Guide

5.13 Develop Architecture Review Board (ARB) Document

Responsibility

Project Management Team

Applicability

The ARB is a review tool.

Description

The mission of the ARB is to review the architectural solutions proposed for major application changes to verify the Department of Human Services (DHS) strategic vision is maintained, existing solutions are leveraged, lessons learned are communicated and the proposed technical solution is supported and understood by all of DHS.

The Objectives of the ARB are:

- To review and approve proposed changes to the technical architecture of a project
- To ensure existing reusable components/solutions are used
- To take advantage of Information Technology (IT) investments
- To confirm adherence to IT standards/best practices

Outputs

The ARB is documented in accordance with project design standards. Maintain the ARB in the Project Folder.

Review Process

Conduct a structured walkthrough to verify the conversion requirements are correct and complete. A representative from the security group is required to attend the walkthrough.

Template

Architecture Review Board (ARB) Document

Reference/Resource

Structured Walkthrough Process Guide

5.14 Conduct Phase Review

5.14.1 Conduct Peer Reviews

Peer reviews are conducted within the project management team to review each of the Requirements Definition Phase deliverables. This is not a formal process but rather a review to ensure team members at the same level agree to the deliverables produced and to leverage the knowledge and experience of others.

5.14.2 Conduct Structured Walkthroughs

Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of outputs and deliverables. The walkthroughs are scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage determines the frequency of the walkthroughs.

Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants. In some cases, it may be beneficial to include software users in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same. For further information, refer to the Structured Walkthrough Process Guide.

5.14.3 Conduct In-Phase Assessments (IPA)

If an IPA is required for this project, the work will be performed by an independent reviewer. The reviewer delivers the IPA report to the project management team which maintains the document in the Project Folder. For further information, refer to the In-Phase Assessment Process Guide.

5.14.4 Conduct Phase Exit

A Phase Exit is conducted at the end of each phase of development or maintenance. The process ends with the receipt of concurrence from the designated approvers to proceed to the next phase. Concurrence indicates all known issues have an acceptable plan for resolution. The project management team maintains the document in the Project Folder. For further information, refer to the Phase Exit Process Guide.

6.0 Detailed System Design (DSD)

Applicability

In the DSD it is important to fill in detail and to incorporate technical information that is required and necessary to automate the business process using data communications, hardware and software. Use this section for all but the simplest express projects.

Description

The user-oriented General System Design (GSD) is now translated into a technical, computer-oriented DSD. Data structures and processing are designed to the level of detail necessary to begin coding. General module specifications define what the modules are to accomplish. Effort focuses on specifying individual routines and data structures while holding constant the software structure and interfaces developed during the GSD. When structural problems are found in the general design, the deficiencies must be addressed before proceeding.

> *Note: Each module and data structure will be considered individually during detailed design, with emphasis placed on the internal and procedural details.*

The primary output of this phase is a DSD document providing a blueprint for the coding of individual modules and programs.

Inputs

Inputs include the:

- Requirements Traceability Matrix *(expanded)*
- General System Design (GSD) Document
- Acquisition and Installation Plans
- System Architecture Document
- Capacity Plan
- Training Plan
- Conversion Plan
- Test Plan

High Level Activities

This section is divided into sub-sections describing the high-level activities performed during this phase. The activities represent the minimum requirements for a large software development effort. Notes are provided to assist in customizing the lifecycle phase requirements to accommodate project scalability. The high-level activities for the DSD Phase consist of:

6.1 Select System Architecture

6.2 Develop Detailed System Design (DSD)

6.3 Refine Data Requirements

6.4 Design Physical Data Model

6.5 Manage Requirements

6.6 Select Programming Standards

6.7 Refine Test Plan

6.8 Develop Proof of Concept/Prototype

6.9 Refine Conversion Plan

6.10 Develop Electronic Commerce Security Assessment (ECSA)

6.11 Conduct Phase Review

Outputs

Several outputs are produced during this phase. The outputs listed below are the minimum requirements for a large software project. The size and complexity of the project determine the extent that additional documentation will be required. Explanations of the outputs are provided under the applicable activities described below:

- Detailed System Design (DSD) Document
- Requirements Traceability Matrix (expanded)
- System Architecture Document (revised)
- Conversion Plan (revised)
- Electronic Commerce Security Assessment (ECSA) Document
- Test Plan (revised)

A matrix showing the outputs associated with each high-level activity is provided in Exhibit 6 0-1, Detailed System Design High Level Activities and Outputs.

Review Process

Quality reviews are necessary during this phase to validate the various outputs. The activities appropriate for quality reviews are identified in this phase and 2.3 Quality Reviews. The reviews are an important milestone in the design process. The time and resources needed to conduct the walkthroughs and Critical Design Review are indicated in the project resources, schedule, and Work Plan Standard.

Reference/Resource

Section 2.3 Quality Reviews, provides an overview of the types of reviews to be conducted within a project.

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

Checklist

Detailed System Design Phase Checklist

Exhibit 6.0-1 Detailed System Design High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
6.1	Select System Architecture	System architecture recommendation	N
6.2	Develop Detailed System Design	Detailed System Design Document	Y
6.3	Refine Data Requirements	Detailed System Design Document (Data Dictionary)	Y
6.4	Design Physical Data Model	Detailed System Design Document (Physical Data Model)	Y
6.5	Manage Requirements	Requirements Traceability Matrix (*expanded*)	Y
6.6	Select Programming Standards	Programming Standards	N
6.7	Refine Test Plan	Test Plan (revised)	Y
6.8	Develop Proof of Concept/Prototype	Proof of Concept	N
6.9	Refine Conversion Plan	Conversion Plan (*revised*)	Y
6.10	Develop Electronic Commerce Security Assessment (ECSA)	Electronic Commerce Security Assessment (ECSA) Document	Y
6.11	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is an output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

6.1 Select System Architecture

Responsibility

Project Management Team

Applicability

This section is applicable when a system architecture has not already been selected and one must be selected or the existing one in which this application will reside must be modified or updated.

Description

When the system architecture for the software has not been pre-determined by the existing computing environment or by the project sponsor and users, architectural alternatives are evaluated to determine the best, cost-effective solution to satisfy the project requirements.

> *Note: "Cost effective solution" does not mean the least expensive alternative. The best, cost effective solution is the alternative doing the best job of satisfying project requirements, ensuring the highest quality software, and providing an adequate return on investment in a timeframe acceptable to the project sponsor.*

Base the selection of specific hardware, software, database management system, and communication facilities on:

- Departmental architecture guidelines or standards
- Hardware and software emphasizing simplicity, flexibility, ease of operation and maintenance
- Cost to procure and maintain potential environment
- Data backup/restore and disaster recovery procedures
- Selection of a distributed or centralized processing environment
- Communication requirements
- Data configuration
- Capacity requirements

Contact the functional support areas or domain experts to aid in the architecture evaluation process. Consultation and input from system and database administrators, local area network administrators, operations personnel, data security staff, developers, and communication experts are helpful.

Outputs

Any output here should be incorporated within the Detailed System Design Document.

Reference/Resource

The Software Development Repository will contain links to any applicable architecture standards.

Tasks

The tasks for selecting the System Architecture consist of:

6.1.1 Evaluate System Architecture Alternatives

6.1.2 Recommend System Architecture

6.1.1 Evaluate System Architecture Alternatives

Responsibility

Project Management Team

Applicability

This section is applicable when the selection of a system architecture is undertaken by the project team.

Description

Factors to consider in an evaluation of architectural alternatives include:

- The experience that the project team members have with each alternative
- The availability of reusable components to facilitate the implementation
- The requirements
- Organizational standards
- The strategic plans for information systems
- The budget

Issues to be considered when evaluating the alternatives:

- Identification of functions to be automated and those that will be manual – including an examination of *what* the automated portion(s) will encompass
- The technical solution for the objectives. The determination of *how* the software is to be designed (e.g., online vs. batch, client-server vs. mainframe, object oriented vs. waterfall approach, web vs. desktop client, Oracle vs. Sybase)
- The project sponsor and users' computing environment and the needs created by the technical solution. Consider any hardware and software to be acquired, including system software and database management products.

One approach for evaluating architecture alternatives is:

- Conduct a Cost Benefit Analysis to determine the most cost-effective alternative. On the benefits side, include the improvements over the current process being used to support the business application. On the cost side, include any degradation from current capabilities and the rationale for allowing the degradation.
- Generate and evaluate a data flow diagram for each alternative.
- Identify how users would interact with the features associated with each alternative (such as the generation of queries and reports).
- List the risks associated with each alternative and develop a plan for mitigating each risk.
- Compare the performance capabilities of each alternative. For example, *"how fast will each alternative be able to process the user's work given a particular hardware resource?"* Performance is usually expressed in terms

of throughput, run time, or response time. Five factors frequently affecting performance include:

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

- Compare the security and access control features of each alternative, including the extent to which each alternative provides protection against human errors, machine malfunction, or deliberate mischief. Common controls include:
  - Check digits on predetermined numbers
  - Batch control totals
  - Creation of journals and audit trails
  - Limited access to files
- Compare the ease with which each alternative allows the system to be modified to meet changing requirements, such as:
  - Fixing errors
  - Changing user needs
  - Mandatory/statutory modifications
  - Enhancements

Outputs

Records for each alternative evaluated are maintained in the Project Folder and used to develop a summary of the system architecture alternatives. The summary will be integrated into the materials presented to the project sponsor when a system architecture recommendation is made.

If a Cost Benefit Analysis is conducted, prepare a report describing the process used for the analysis, a summary of the alternatives considered, and the results obtained and maintain the report in the Project Folder. The report will be integrated into the materials presented to the project sponsor when a system architecture recommendation is made. Review Process

Structured walkthroughs are conducted to review the records for each alternative.

Reference/Resource

Structured Walkthrough Process Guide

6.1.2 Recommend System Architecture

Responsibility

Project Management Team

Applicability

This task is necessary when additional resources are required, when deviations with organizational standards are required and when seeking buy-in from future project team members or stakeholders.

Description

Based on the results of the architecture alternatives evaluation, a recommendation is developed for a cost-effective system architecture satisfying the project requirements.

Prepare a presentation for the project sponsor and users providing the supporting information:

- The limitations or problems with any current manual or automated process to be resolved by the software.
- The logical model for the software. Highlight new functionality to be implemented.
- The information for each architecture alternative evaluated, to include:
  - A description of the alternative
  - An overall data flow diagram showing how the alternative would be implemented
  - The way the system would look to the users, in terms of hardware, user interface, reports, and query facilities
  - The estimated benefits of the alternative
  - The estimated cost and time to implement the alternative
  - A statement of the element of risk associated with the alternative
- The recommended alternative and explanation for the selection.

Concurrence must be reached from all stakeholders regarding the selection of the system architecture before proceeding to the next phase of development. Any delay in making this decision may result in missed deadlines established within the project schedule.

Outputs

Document the recommended system architecture and a summary for each alternative in the System Architecture Document. Include the recommendation of any background information relevant to the decision process, such as a Cost Benefit Analysis Report. Describe the rationale for proposing the recommended architecture along with an analysis of the impact on the users' organization. Include the impact analysis for existing and other planned systems.

Present the architecture recommendation to the project sponsor and users, either as a presentation or a distributed document. Maintain the document in the Project Folder.

Review Process

A structured walkthrough is conducted to review the completeness and accuracy of the evaluations and recommendation.

Reference/Resource

Structured Walkthrough Process Guide

6.2 Develop Detailed System Design (DSD)

Responsibility

Project Management Team

Applicability

This DSD should be created for any substantial new system or significant modification to an existing one where the general design is inadequate in detail to proceed with system development.

Description

The DSD is the main deliverable from the DSD Phase. Before coding and testing begins, it translates the requirements into precise descriptions of the software components, interfaces, and data. The DSD is the blueprint for the Development Phase and is based on the software structure and data model established during the General System Design (GSD) Phase.

After the project sponsor's formal approval of the DSD Document, the system design is baselined. Once the DSD has been baselined, future changes must be managed under the change control procedures established in the IRM Standards/Business and Technical Standards/Operations Support Domain/Configuration and Change Management. Records of all approved changes are incorporated into the DSD Document.

> *Note: It is important for the project sponsor and users to understand any changes to the baselined DSD. This may affect the project scope, cost, required resources, and the project schedule.*

The Project Management Team is responsible for the identification of any requested changes resulting in a change to the project's scope. Evaluate the potential impact of any changes to the project cost, resources, or project schedule. Notify the project sponsor when revisions are required to accommodate the change requests.

Each requirement in the Requirements Definition Document (RDD) must be traceable to one or more design entities. This traceability ensures the software satisfies all the requirements and does not include inappropriate or extraneous functionality.

Expand the Requirements Traceability Matrix developed in the Requirements Definition Phase to relate the DSD to the requirements.

Outputs

Maintain the Detailed Design Document and the Expanded Requirements Traceability in the Project Folder.

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix.

Refer to task 6.2.2 Conduct Critical Design Review, for the DSD review process.

Reference/Resource

Structured Walkthrough Process Guide

Tasks

The tasks for developing the DSD consist of:

- 6.2.1 Develop Detailed System Design (DSD) Document
- 6.2.2 Conduct Critical Design Review
- 6.2.3 Develop Program Specifications

6.2.1 Develop Detailed System Design (DSD) Document

Responsibility

Project Management Team

Applicability

This section is applicable whenever a detailed system design document is created. A DSD should be developed for all projects except express projects. See 6.2 Develop Detailed System Design (DSD) - Applicability.

Description

The DSD Document is a description of the software structures, components, interfaces, and data necessary to support the programming process. The design is based on the requirements found in the Requirements Definition Document (RDD) and must be consistent with and expand upon the framework provided in the general systems design (GSD) document.

The DSD Document is submitted to the project sponsor and users for review and approval. The approved design document is the official agreement and authorization for designing and building the software. Approval implies the design is understood, complete, accurate, and ready to be used as the basis for subsequent lifecycle phases.

The approved document specifies the baselined design. Subsequent changes or additions must receive stakeholder concurrence and establish a revised design. This document should be considered a “living” document throughout the remainder of the SDM, but subject to the change control management process.

Outputs

Maintain the approved DSD Document in the Project Folder.

Review Process

Conduct structured walkthroughs to ensure the DSD Document is accurate and complete.

After the completion of the DSD Document, schedule an In-Phase Assessment (IPA) if required.

Template

Detailed System Design (DSD) Document

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

6.2.2 Conduct Critical Design Review

Responsibility

Project Management Team

Applicability

This step will always be performed whenever a formal DSD is created.

Description

The Critical Design Review is a formal technical review of the Detailed System Design (DSD). The purpose of the review is to show the document is consistent with the requirements document and the GSD and that it adequately addresses the informational needs required during the development process. The review must show that all features must be traceable back to the requirements document and that all requirements have been included. In addition, all design constraints (e.g., performance, interface, security, resource, and reliability requirements) are encompassed by the proposed design.

The level of technical detail presented in the review will depend on many factors, including the audience. The meetings are usually attended by the project sponsor, users, and on occasion, a technically oriented audience. At times, the validity of algorithms used to perform critical functions must be addressed.

Note: Several short Critical Design Reviews can replace one long review if the software components are not highly interdependent.

The Critical Design Review is conducted to determine whether or not the design specifications are capable of supporting the full functionality of the software. The review includes a determination of whether or not:

- The algorithms will perform the required functions
- The specification is complete, unambiguous, and well documented, including inputs, outputs, process description, interfaces, timing, sizing, and data storage allocations
- The specification is necessary, sufficient, and directly traceable to the requirements definition document (RDD)
- The specification is compatible with every other related specification, piece of equipment, facility, and item of system architecture, especially regarding information flow, control, and sequencing
- Each component specification is consistent with the abilities of current development and user personnel

In addition to verifying individual specifications, the Critical Design Review assesses other project outputs to ensure:

- The team is following the approved design approach
- Adequate measures are taken to reduce risk on a technical, cost, and schedule basis
- The performance characteristics of the design solution are acceptable

- Testing will be sufficient to ensure software correctness
- The resulting application will be maintainable
- Provisions for automatic, semi-automatic, and manual recovery from hardware/software failures and malfunctions are adequate and documented
- Diagnostic programs, support equipment, and commercial manuals comply with the system maintenance concept and specification requirements

Outputs

Official meeting minutes for each design review session are distributed to all participants. The minutes consist of significant questions and answers, action items of the individual/group responsible, deviations, conclusions, and recommended courses of action resulting from presentations or discussions. Recommendations not accepted must be documented indicating the reason for non-acceptance. The project sponsor (or user representative) determines review performance in one of three ways:

- *Approval* - The review was satisfactorily completed.
- *Contingent Approval* - The review is not finished until the satisfactory completion of resultant action items.
- *Disapproval* - The specification is inadequate. Another Critical Design Review will be required.

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

6.2.3 Develop Program Specifications

Responsibility

Development Team

Applicability

Specifications are required for all programs, modules and classes.

Description

Program Specifications are the written procedural descriptions of each software component. The Program Specifications provide precise and adequate information necessary for the application staff to develop the code. A consistent method of creating specifications should be adhered to for all components.

Many techniques are available for creating the program specifications, including the use of:

- Formal specification languages
- Program design languages – e.g., pseudocode or structured English
- Metacode, tabular tools – e.g., decision tables
- Graphical methods – e.g., flow charts or box diagrams

Teams on projects using object-based and Object-Oriented (OO) methods may use a different approach. Often modules and components are developed in small groups with specifications being prepared followed by coding without producing all of the specifications prior to proceeding with development. Development then proceeds in an iterative fashion. In many O.O. cases, the requirements specification, preliminary design constraints and dependencies, often results in detailed specifications formulated in the design language.

The team members may select one or more techniques in determining the suitability of the software, along with the requirements of the team members using the DSD.

Outputs

Software specifications may be produced as documents, graphic representations, formal design languages, records in a database management system, and the Computer Aided Software Engineering (CASE) tool dictionaries.

Sample Attributes

For each program to be constructed, the functional and technical attributes are documented. Typical attributes include:

- Program identification
- Program name
- Program purpose
- Program generic type
- Functional description

- Program hierarchical features diagram
- Development dependencies and schedule
- Operating environment
  - equipment
  - programming language and version
  - preprocessor
  - operating system
  - storage restrictions
  - security
- Frequency of use
- Data volumes
- Program termination messages
  - normal termination
  - abnormal termination
- Console/printer messages
- Recovery/restart procedures
- Program input/output diagram
- Data bank information
- Called and calling programs/modules with defined parameters
- Program logic diagrams
- Significant "how-to" instructions
- Telecommunications information
- Recommended coding techniques

Review Process

One or more structured walkthroughs are recommended to ensure Program Specifications are accurate and complete.

Reference/Resource

Structured Walkthrough Process Guide

Program specifications within the DSD Document template

6.3 Refine Data Requirements

Responsibility

Development Team, Data Analyst, and Data Administration

Applicability

This section is applicable whenever the data dictionary will require updating beyond what has already been done for this project.

Description

The data dictionary is finalized in the Detailed System Design (DSD) Document to reflect changes noted in the DSD phase. The Data Dictionary was developed in the Requirements Definition and General System Design (GSD) phases.

Outputs

The data dictionary is reviewed and updated to complete the detailed information on data elements and their physical characteristics, data conversion requirements, and business information.

Review Process

Data Administration conducts a structured walkthrough to verify the data dictionary is correct and complete.

Reference/Resource

Enterprise Data Dictionary

Structured Walkthrough Process Guide

6.4 Design Physical Data Model

Responsibility

Development Team and Data Analyst

Applicability

Whenever data storage structures for data movement structures will be added or modified.

Description

The physical data model is a description of the dynamics, data transformation, and data storage requirements of the software. To achieve a specific technical solution, the logical data model developed during the General System Design (GSD) phase is used to build a physical data model. Retain all the constructs inherent in the logical data model, with the exception of the naming constructs which must reflect the appropriate standards. If the physical data model is not consistent with the logical data model, the differences must be reconciled by updating the general design, the detailed design, or both before proceeding.

The physical data model typically differs from the logical data model in the:

- Key information – Primary and Foreign keys are not shown in the logical data model
- Constraints imposed by the database management system – The logical data model may have different implementations in the selected database management system.
- Performance – Data redundancies, indices, and data structure changes may have to be introduced into the physical data model to improve performance.
- Distributed processing – Possible network and multiple production hardware configurations may cause changes to the physical data model.

During database design, develop a description of the required implementation for the master and transaction files. The features of the designed database are to include:

- Report writer and file processing capabilities
- Online retrieval, updating, and storage of data
- Automated data dictionary systems
- Extensible Markup Language (XML) where applicable

Outputs

The physical data model is documented, maintained in the Project Folder, and incorporated into the Detailed System Design (DSD) Document.

Review Process

Structured walkthroughs are conducted to verify the physical data model is correct and complete.

Reference/Resource

Structured Walkthrough Process Guide

Instructions for Completing the Data Dictionary Excel Spreadsheet

ITP-INF003 Data Modeling Standards

6.5 Manage Requirements

Responsibility

Project Management Team

Applicability

This is to be used in any development effort where scale warrants the coordination of resources including staff, budget, and procurement. Generally the more stakeholders involved the more essential this task becomes.

Description

At the completion of the Detailed System Design (DSD) Document, review and cross-reference all the requirements in the DSD and the Requirements Traceability Matrix.

Each requirement identified in the Requirements Definition Document (RDD) must be traceable to one or more design entities. This traceability ensures the software satisfies the requirements and does not include inappropriate or extraneous functionality. The Requirements Traceability Matrix developed in the Requirements Definition Phase and further updated during the General Design phase is expanded to relate this Detailed System Design (DSD) to the requirements.

Outputs

Maintain the expanded Requirements Traceability Matrix in the Project Folder.

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix to ensure all required functionality is included.

Reference/Resource

Structured Walkthrough Process Guide

6.6 Select Programming Standards

Responsibility

Project Management Team and Development Team

Applicability

This step is applicable where there is not an official standard in place. The team should select a standard based upon best practices in government and industry. It is also recognized that standards can become obsolete in whole or in part, so it may be necessary and proper to request revisions to existing standards. Selection of a standard usually will not be necessary.

Description

Programming standards are necessary to ensure custom-built software has acceptable design and structural properties. Departmental standards help with consistency between various projects throughout the organization.

When it is necessary to select a standard given that one does not exist, the team should select one based upon best practices in government and industry. This standard can be updated to address reoccurring or likely problems that occur within the Department. This standard or updates to it then must be submitted to the application domain and approved by them.

Any deviation from established standards must be approved and documented in the Project Plan. The selected standard and the development of a new standard needs to include:

Control Flow Constructs

- Sequence
- If-then-else
- Case statement
- Do-while (pretest loop)
- Do-until (post-test loop)

Module Size

- As a “rule-of-thumb” the number of executable lines of source code should average 100 lines per unit.
- As a “rule-of-thumb” the units should contain no more than 200 lines of executable source code.

Module Design

- Units do not share temporary storage locations for variables
- Units perform a single function
- Avoid self-modifying code
- Each unit is uniquely named
- Each unit has a standard format:
    - Prologue
    - Variable declarations
    - Executable statements/comments
- Use single entry/exit points except for error paths
- Set comments off from the source code in a uniform manner

Symbolic Parameters

- Use instead of specific numeric values
- Use for constants, size of data structures, relative position in list

Naming Conventions

- Use uniform naming throughout each unit and module to be put under configuration control
- Use meaningful variable names
- Do not use keywords as identifiers

Mixed Mode Operations

- Avoid mixed mode expressions
- Add comments in code whenever used

Error and Diagnostic Messages

- Design messages to be self-explanatory and uniform
- Do not require user to perform table lookups

Style

- Use indentation, white space, and blank lines to enhance readability
- Align compound statements
- Avoid “goto” statements.
- Avoid compound, negative Boolean expressions
- Avoid nesting constructs beyond five levels deep

- Avoid deeply nested “if” statements.
- Use parentheses to avoid ambiguity
- Include only one executable statement per line
- Avoid slick programming tricks that may introduce defects and are difficult to maintain – the most direct solution is best.

Outputs

Identify and distribute the programming standard to the project management team members.

Review Process

Conduct a peer review to ensure the programming standards are complete and appropriate for the project’s programming language and tools. When new standards are used, they should be recommended as standard revisions when appropriate.

Reference/Resource

IRM Standards/Business and Technical Standards/Application Domain

ITB-APP001 Business Solutions Center of Excellence (BSCoE)

6.7 Refine Test Plan

Responsibility

Project Testing Team

Applicability

The test plan should always be reviewed after or during the development of the Detailed System design (DSD). Whenever, the current test plan needs to be expanded upon to prove the validity of executable code, this step is applicable. For most medium to large projects this step will be necessary.

Description

This will include integration testing, system testing, user acceptance testing and all variations and forms of it. Testing of these types can take the form of parallel tests and pilot tests. Generally, test plan refinement is necessary to prove the validity of software and all of its functionality as it is specified in the DSD. Refinement of the test plan might require detailing generalized tests, updating any schedules and adding tests that were previously not included. Tests defined become more specific and when test case scenarios are used, they are likely to be expanded upon to validate the processes. Tests that are likely to be added or detailed include:

- Computations
- New functions
- Common components defined during the DSD
- Look and feel tests
- Navigational Tests
- Verification of Inputs and Outputs
- Verification of presentation and format
- Workflows
- Software component Interfaces

Refine the Test Plan to describe the testing effort, to provide the testing schedule, and to define the complete range of test cases to ensure the reliability of the software. Test cases must be complete and the desired outputs known before testing is started. The test plan addresses:

- A definition of, and the objectives for, each test case
- Definition of the test scenario(s) including the step-by-step procedure, the number of processing cycles to be tested or simulated, and the method and responsibility for loading test data into the system

- Specifications of the test environment including the hardware and software environments under which the testing are to be conducted. Identify and describe manual procedures, automated procedures, and test sites (real or simulated)
- Identification of test tools and special test support needs (e.g., hardware and software to simulate operational conditions or test data recordings of live data)
- Identification of responsibilities for conducting tests, reviewing, reporting, approving the results, and correcting error conditions
- A requirements verification matrix mapping individual tests to specific requirements, and a description of how each system requirement is to be validated
- A schedule for integrating and testing all components, including adequate time for re-testing

Integration Testing

Integration testing is used to verify the integrity of a module and interfaces with other modules within the software structure. The Integration Test Plan is developed to incorporate modules (successfully tested) into the overall software structure and to test each level of integration to isolate errors.

Except for the largest or most critical projects, the Integration Test Plan is part of the Project Test Plan.

The number of integration levels, the classes of tests to be performed, and the order of routines and builds are incorporated into the overall software structure and addressed in the Integration Test Plan. Questions to be considered when developing the plan:

- Will routines be integrated in a purely top-down manner or should builds be developed to test sub-functions first?
- In what order should the major software functions be incorporated?
- Is the scheduling of module coding and testing consistent with the order of integration?
- Is special hardware required to test certain routines?

Include Integration Testing to validate:

- Interfaces between the module and all other modules
- Each input message or command processed by the module
- Each external file or data record referenced by coding statements in the module
- Each message, display, or record generated by the module

An important consideration during integration test planning is determining the amount of test software development is required to perform adequate testing. For example:

- It may be cost-effective to delay testing of a communication function until hardware is available rather than generate test software to simulate communication links.
- Include certain completed modules within the software structure to avoid having to develop software drivers.

Base the decisions on the cost and risks.

System Testing

The goal of the System Test plan is to complete system tests on schedule and within project constraints. The purpose is to prove and validate overall system functionality. Perform system testing to ensure the software:

- Adequately satisfies the project requirements
- Functions in the computer operating environment
- Successfully interfaces between user procedures, operating procedures, and other systems
- Protects the software and data from security risks

As much as possible, test the system under the same daily conditions encountered during regular operations. Test the system timing, memory, performance, and security functions to verify they perform as specified. In addition, test for verification under normal and high-load conditions by the functional accuracy of logic and numerical calculations.

Test data, varied and extensive, is to enable the verification of the operational requirements. In the test plan include the expected output results in the form of calculated results, screen format, hardcopy output, predetermined procedural results, warnings, error messages, and recovery.

Detailed planning for system testing helps to ensure the system acceptance will be successfully completed on schedule. System testing includes:

- Performance tests measuring throughput, accuracy, responsiveness, and utilization under normal conditions and at the specified maximum workload
- Stress tests to determine the loads resulting in appropriate, non-recoverable, or awkward system behavior
- Interface tests to verify the system generates external outputs and responds to external inputs as directed by approved interface control documentation
- System recovery and reconfiguration tests
- Verification of the system can be properly used and operated in accordance with the user’s guide and operating instructions
- Verification the system meets the requirements for reliability, maintainability, and availability, including fault tolerance and error recovery

- Verification of the effectiveness of error detection and analysis, and automated diagnostic tools
- Demonstrate the system complies with the serviceability requirements, such as accessibility, logistics, upgrades, diagnostics, and repair capabilities

Pre-SAT Testing

Pre-SAT can be considered an extension of systems integration testing or preliminary requirement prior to user acceptance testing. The emphasis of Pre-SAT is two-fold: 1) Ensure all systems components connectivity and interfaces are configured and work properly, and 2) Preliminary configuration and preparation to facilitate proper user acceptance test execution. The goal is to ensure all the systems components are configured, connected, and work properly as well as establish an effective and efficient user acceptance test environment for the end users.

Acceptance Testing

Acceptance Testing is used to thoroughly exercise and validate the systems functionality as well as a primary component to certifying operational readiness. However, the emphasis of acceptance testing is to include the actual system users and application owners. When this step is defined separately for system testing, this step will follow the systems test. The purpose is to prove functionality and acceptability to the end user. The goal is to gain user signoff and approval to proceed with implementation activities and planning.

Outputs

Maintain the refined test plan in the Project Folder.

Review Process

Conduct peer reviews or structured walkthroughs, as needed, to ensure each system test procedure is accurate, complete, and accomplishes the stated objectives. Review and revise the initial System Test Plan, as needed, during the Development Phase.

Reference/Resource

Structured Walkthrough Process Guide

6.8 Develop Proof of Concept/Prototype

Responsibility

Project Management Team

Applicability

This section applies whenever a prototype is considered useful and practicable. A prototype/proof of concept can serve numerous purposes. One is most useful when it is difficult to get a clear description from the user/project sponsor of their expectations or requirements from an automation project. Prototypes are also quite useful in demonstrating what is or what is not possible. Finally this step is useful in validating what has been communicated, thus minimizing risks when proceeding with a costly development effort.

Description

Proof of Concept/prototype is developed to test and demonstrate the architecture. There are different types of prototypes prepared. One or more prototypes may be chosen within the project. Reviews are conducted, if prototypes are built. The type of prototype or the creating of a proof of concept/prototype is dependent on the project to demonstrate feasibility of the whole, parts, or certain unique aspects of the application. A prototype helps close the communications gap between the end user, project sponsor, and the development team. In addition, a prototype can facilitate the essential participation of the end user in the project. It mitigates the likelihood of project failure due to the delivered application not meeting the end users expectations.

Types of Prototypes

- **Behavioral prototype -** focuses on exploring specific behavior of the system
- **Structural prototype -** explores some architectural or technological concerns
- **Exploratory prototype -** thrown away when done, also called a throwaway prototype
- **Evolutionary prototype -** gradually evolves to become the real system

Refer to the guidelines for developing the proof of concept in 7.3 Write Programs.

Outputs

The outputs for the Proof of Concept/Prototype is the completed units and modules of code based on the type of prototype developed.

Review Process

Conduct a structured walkthrough of the Proof of Concept/Prototype to ensure that it is consistent with user requirements and that the intended purpose of the prototype is met.

Reference/Resource: Structured Walkthrough Process Guide

6.9 Refine Conversion Plan

Responsibility

Project Management and Development Teams

Applicability

This section is applicable only if a pre-existing system's database is being converted or the new database will be constructed from other data sources.

Description

The Conversion Plan created in the General System Design (GSD) Phase is modified based on new information gathered in the Detailed System Design (DSD) Phase. If the software replaces an existing data store or populated through one, a conversion plan must be created.

When not already done, identify the conversions needed, how the conversions are to be implemented, how the conversion process is to be tested, the conversion schedule, and how the conversion is to be implemented in the Conversion Plan. When appropriate, add reports detailing the rejected data as a result of the data cleansing/purification procedure.

All data mappings, conversions, data cleansing procedures, computations, and derivations must be detailed. Manual processes must be described procedurally and conversion software components must have specifications. A test plan is important to validate the quality on the converted data and to provide feedback to the project sponsor. It can never be assumed a conversion will be run once. It may be necessary to run it during development, system testing for acceptance testing, and finally prior to final implementation. Therefore, a conversion schedule is important to be consistent with and coordinated with the project schedule. Include a conversion report or process summary for the project stakeholders and validate the conversion process results.

Outputs

Maintain the fully detailed conversion plan, complete with updated schedule, manual procedures, and program specifications in the Project Folder. The plan should be submitted to the project manager or management team for inclusions within the project schedule.

Review Process

Conduct structured walkthroughs, as needed, to ensure the Conversion Plan is accurate and complete. The project management team should review the conversion schedule against the existing project schedule.

Reference/Resource

Structured Walkthrough Process Guide

6.10 Develop Electronic Commerce Security Assessment (ECSA)

Responsibility

Project Management Team

Applicability

This section is applicable whenever data movement occurs beyond the Departments firewall. This would include all internet applications and data file transfers between the department and any external entity.

Description

Prior to participating or initiating an electronic transaction, an executive agency shall complete and submit an Electronic Commerce Security Assessment (ESCA) to the Office of Administration/Office of Information Technology (OA/OIT). The ESCA evaluates the proposed use, transmission, or storage of the electronic record, and recommends the security procedures to be used based upon certain criteria (i.e., the intended use of the electronic record, type of information being transmitted, received or stored, degree of risk to the Commonwealth, and users of the system, and potential legal liability).

The purpose of the ECSA is to assist program managers, information technology professionals, and senior agency leadership in determining the appropriate level of security necessary for sending, receiving, and processing electronic transactions. It is designed to provide a structured and consistent method for making such determinations. In addition, it provides guidance regarding the level of security designed into e-government applications.

Beyond providing the Office of Administration/Office of Information Technology (OA/OIT) with the information it requires to comply with the Electronic Transactions Act, the ESCA serves a more specific purpose to the Department of Human Services. The Assessments are a key decision driver in the Department's information security risk management activities. The Assessments drive budgetary, technical infrastructure, and resource allocation decisions throughout the Department. Therefore, program managers, information technology personnel, and agency executive staff need to approach the Assessments with the appropriate level of diligence.

Outputs

Submit the ECSA document via email to the Department Security Officer for Department review and subsequent submission to OA/OIT. Maintain the ECSA in the Project Folder.

Review Process

Submit the ECSA for review and approval to the Configuration Management Section/Service Level Management Unit. Once the internal approval is granted, the document will be submitted on your behalf to the OA/OIT. Applications will not be allowed to go-live until an ECSA has been approved. The approximate time required for review and approval is five weeks once submitted to OA/OIT. For complete detailed instructions, refer to the OA/OIT site at Electronic Commerce Security Assessment Application.

Reference/Resource

IRM Standards/Business and Technical Standards/Security Domain/OA/OIT Security Policies

6.11 Conduct Phase Review

6.11.1 Conduct Peer Reviews

Peer reviews are conducted within the project management team to review each of the Requirements Definition Phase deliverables. This is not a formal process but rather a review to ensure team members at the same level agree to the deliverables produced and to leverage the knowledge and experience of others.

6.11.2 Conduct Structured Walkthroughs

Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of outputs and deliverables. The walkthroughs are scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage determines the frequency of the walkthroughs.

Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants. In some cases, it may be beneficial to include software users in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same. For further information, refer to the Structured Walkthrough Process Guide.

6.11.3 Conduct In-Phase Assessments (IPA)

If an IPA is required for this project, the work will be performed by an independent reviewer. The reviewer delivers the IPA report to the project management team which maintains the document in the Project Folder. For further information, refer to the In-Phase Assessment Process Guide.

6.11.4 Conduct Phase Exit

A Phase Exit is conducted at the end of each phase of development or maintenance. The process ends with the receipt of concurrence from the designated approvers to proceed to the next phase. Concurrence indicates all known issues have an acceptable plan for resolution. The project management team maintains the document in the Project Folder. For further information, refer to the Phase Exit Process Guide.

7.0 Development

Applicability

The Development Phase, subsequent to all the documentation from the Detailed Design Phase being prepared, reviewed and approved; the detail designs are translated into executable software with multi-level testing integrated throughout the evolution of the software coding construction (i.e., unit, module, systems integration, and user acceptance outline in Test Phase in **Section 8**). The objectives are to ensure the systems function as expected and user requirements are satisfied. Software testing must be integrated into each progressive evolution of the software code construction and executed at every level of product development. Comprehensive test plans must be establish to ensure effective test coverage and effectiveness, detect and correct errors up stream, verify end product quality, and validate systems functionality and operational readiness. Test scenarios and their respective data sets must be derived and their correctness and consistency should be monitored throughout the development process with clear traceability to both functional and non-functional systems requirements. The multi-level test strategies and test plans must be outlined along with their respective results documented throughout the software coding construction in order to realize a quality end product.

Description

This phase focuses on the transformation of the Detailed System Design (DSD) into working software that is tested throughout the software code construction evolution with validation check points to ensure end product is compliant with all predefined functional and non-functional requirements (i.e., business operational and systems technical specifications). Any hardware or software procured to support the programming effort is installed.

Generate the source code, including suitable comments, using the approved program specifications. Code the supporting modules, including any database utilities. Group the source code into process able units, and compile all high-level language units into object code. Perform multi-level testing (i.e., unit, module, systems integration, and user acceptance) to determine if the code satisfies the program specifications and is complete, logical, error free, fully functional, reliable, and validated/certified for release into live production environments to support business operations.

In addition, produce the testing and operating documentation required for validating/certifying operational readiness, configuring, installing, operating, and supporting the software through its lifecycle. Design a training program and a Training Plan describing the program. Develop plans for the acquisition and installation of the operating environment hardware and software.

Inputs

Inputs include the:

- Multi-level Software Test Strategies
- Requirements Traceability Matrix *(expanded)*

- Detailed System Design (DSD) Document
- Software Specifications (i.e., functional and non-functional)
- Software Engineering Process and Programming Standards
- Conversion Plan

High-Level Activities

This section is divided into sub-sections describing the high-level activities performed during this phase. The activities represent the minimum requirements for a large software development effort. Notes are provided to assist in customizing the lifecycle phase requirements to accommodate project scalability. The high-level activities for the Development Phase consist of:

7.1 Develop Acquisition Plan

7.2 Establish Programming Environment

7.3 Write Programs

7.4 Conduct Unit Testing

7.5 Establish Allocated Baseline

7.6 Manage Requirements

7.7 Develop Test Plan (Iterative plan aligned with software construction evolution)

a) Test Scenarios (i.e., Unit, Module, Sys Integration, End-User Acceptance)

b) Test Scripts (i.e., Load & Performance, Software Vulnerability, other ancillary tests and/or specific technology and infrastructure component testing)

c) Define Test Parameters, Check Lists, Criteria, Governance, and Reporting

d) Define Resource Requirements (i.e., data sets, environments, staffing, etc.)

e) Define Test Cycles, Durations, and Schedules

f) Cross-matrix links to functional and non-functional requirements

7.8 Develop and Execute Test Scenarios

7.9 Develop and Execute Test Scripts

7.10 Test Document Repository

7.11 Transition to Operational Status

7.12 Generate Operating Documentation

7.13 Refine Training Plan

7.14 Finalize Capacity Plan

7.15 Establish Integration Test Environment

7.16 Develop Batch Operations

7.17 Conduct Phase Review

Outputs

The size and complexity of the project determine deviations in the content and delivery of the outputs. Explanations of the outputs are provided under the applicable activities described in the remainder of this section. Typical requirements of a large development project include:

- Application Code
- Unit/Module Test Scenario Checklist
- Systems Integration Test Scenario Check List
- Requirements Traceability Matrix (*expanded*)
- Test Scenarios
- Test Scripts
- Operating Documentation *(initial)*
- Transition Plan
- Training Plan (*revised*)
- Capacity Plan (*final*)
- Batch Operations Manual (BOM)
- Batch Operations Services Request
- Test Plan (*final; reference 7.7 above*)

A matrix showing the outputs associated with each high-level activity is provided in Exhibit 7.0-1, Development High Level Activities and Outputs.

Review Process

Quality reviews are necessary during this phase to validate the product and associated outputs. The activities appropriate for quality reviews are identified in this phase and 2.3 Quality Reviews. The time and resources needed to conduct the quality reviews should be reflected in the project resources, schedule, and Work Plan Standard.

Reference/Resource

Section 2.3 Quality Reviews, provides an overview of the types of reviews to be conducted within a project.

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

Checklist

Development Phase Checklist

Exhibit 7.0-1 Development High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
7.1	Develop Acquisition Plan	Acquisition Plan	N
7.2	Establish Programming Environment	Physical Installation of Hardware and Software	Y
7.3	Write Programs	Application Code	Y
7.4	Conduct Unit Testing	Unit Test Scenario Checklist Test Plan (*final*) Systems Test Plan	N Y Y
7.5	Establish Allocated Baseline	Internal build test procedures and results	Y
7.6	Manage Requirements	Requirements Traceability Matrix (*expanded*)	Y
7.7	Develop Test Plan	(Iterative plan aligned with software construction evolution, submitted as final article)	Y
7.8	Develop and Execute Test Scenarios	Test Scenarios (i.e., Unit, Module, Sys Integration, End-User Acceptance)	Y
7.9	Develop and Execute Test Scripts/Checklists/Results	(i.e., Load & Performance, Software Vulnerability, other ancillary tests and/or specific technology and infrastructure component testing)	Y
7.10	Test Document Repository	Test Document Repository (links all test plans, scenarios, scripts, and results per software release)	N
7.11	Plan Transition to Operational Status	Transition Plan	Y
7.12	Generate Operating Documentation	Operating Documentation	Y
7.13	Refine Training Plan	Training Plan (*revised*)	Y
7.14	Finalize Capacity Plan	Capacity Plan (*final*)	Y
7.15	Establish Integration Test Environment	Validated Output	Y
7.16	Develop Batch Operations	Batch Operations Manual Batch Operations Services Request	Y Y
7.17	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

7.1 Develop Acquisition Plan

Responsibility

Project Management Team

Applicability

Developing an acquisition plan is essential to create the path for the entire acquisition. The process is structured along various time lines based upon the complexity of the requirements. These time lines are consumed for funding and the timing of contract awarded.

Description

An Acquisition Plan is developed for the required hardware, software, and communications equipment. This process includes the installation and operations requirements for all known user sites and addresses any special procurement necessary to accommodate the equipment at a particular site. Acquisition planning must include sufficient time to accomplish all procurement, delivery, testing, and installation processes.

It may be necessary to perform a risk analysis of the impact of certain resources not being available when needed. Develop a contingency plan for dealing with needed resources acquired later than expected.

Outputs

When an Acquisition Plan is developed, the project sponsor and user representatives ensure all site-specific hardware, software, and communications needs are addressed. Maintain the Acquisition Plan in the Project Folder.

> *Note: The Acquisition Plan may be combined with the Installation Plan for projects not requiring extensive procurement and installation of hardware and software.*

Review Process

Conduct a structured walkthrough, as needed, to ensure the Acquisition Plan is accurate and complete.

Reference/Resource

Structured Walkthrough Process Guide

7.2 Establish Programming Environment

Responsibility

Project Management Team

Applicability

This is the actual installation of the required software and hardware.

Description

This activity involves the assembling and installation of hardware, software, communications equipment, databases, and other items required to support the programming effort. Once the installation of the equipment or software is complete, testing is conducted to verify the operating characteristics and functionality of the hardware and software. If required, activate the security software and procedures when the installations are complete.

> *Note: The production environment is not to be used for development.*

Prior to support of product development, test and verify the vendor products satisfaction of the:

- Product performs as advertised/specified
- Product's performance is acceptable and predictable in the target environment [e.g., testing for Local Area Network (LAN) certification]
- Product fully or partially satisfies the project requirements
- Product is compatible with other hardware and software tools

Time should be allotted for the project management team to become familiar with the new products. Ensure the project management team members using the hardware or software obtain proper training. This may involve training sessions conducted by the vendor or a consultant.

This is a good time to review programming standards selected during Detailed System Design (DSD).

Outputs

Physical installation of the required equipment and testing reports

Review Process

Follow up on the testing reports.

7.3 Write Programs

Responsibility

Development Team and Application Developer's Team

Applicability

Software construction in today's development projects goes beyond the old paradigm of only compiling and linking code into executables. The widespread use of interpreted languages [e.g., scripts and Structured Query Language (SQL) statements] and tagging languages [e.g., Hypertext Markup Language (HTML) and eXtensible Markup Language (XML)] have broadened the scope of this activity. The term "write programs" has been retained and still communicates the idea of "software construction," but, it goes beyond the old paradigm of compile/link/load.

Description

The activity involves generating source and object code, executables, and scripts for the software. Write the source code in accordance with the programming standards selected in the Detailed System Design (DSD) Phase. Regardless of the platform, code development is to adhere to a consistent set of programming techniques and error prevention procedures. This will promote reliable, maintainable code, developed in the most efficient and cost effective manner.

Identify and store the source code and executables in a way to facilitate the configuration control measures described in the IRM Standards/Business and Technical Standards/Operations Support Domain/Configuration and Change Management.

The tasks for writing programs include:

- Using the Program Specifications developed in the DSD Phase as the basis for the coding effort
- Writing, preparing and generating source code, methods, objects, classes, and scripts
- Generating the physical files and database structure
- Generating video screens, report generation codes, and plotting instructions
- If conversion of an existing system or data is necessary, generating the programs described in the Conversion Plan
- Conducting preliminary testing of completed units. When the test output is correct, reviewing the program specification to ensure the unit or module conforms to the specification

Coding Practices

Coding practices recommendations are:

- Meet with the programming staff, as needed, to discuss problems encountered and to facilitate program integration and uniformity
- Use a standardized set of naming conventions for programs, data elements, variables, and files to achieve program uniformity
- Facilitate the development and maintenance of modules being shared by programs, to require the same functionality
- Include internal documentation for each program
- Back up all code on a daily basis and store in an offsite location to avoid catastrophic loss
- Determine a standard format for storing and reporting elements representing numeric data, dates, times, and information shared by programs
- Update the DSD Document to reflect any required deviations from the documented design

Outputs

The outputs produced include:

- Completed units and modules of code
- Test materials generated from preliminary testing

Each requirement identified in the Requirements Definition Document (RDD) must be traceable to the code. This traceability ensures the software will satisfy the requirements and will not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix to relate the requirements to developed software. Maintain the matrix in the Project Folder.

Review Process

Informal reviews of each application developer's work are recommended in order to keep team members informed of progress and to facilitate the resolution of problems. The combined knowledge and skills of the team members helps build quality into the software and support the early detection of errors in design, logic, or code.

Conducted Structured Walkthroughs to review the expanded Requirements Traceability Matrix and complete the modules to ensure the code is accurate, complete, well documented, and satisfies project requirements. For large or complex projects, conduct code inspections at successive phases of code construction.

A code inspection is a static analysis technique relying on visual examination of code to detect errors, violations of development standards, and other problems. The inspections are particularly important when several application developers or different programming teams are producing code. The inspection team may include experts from outside of the project. Ideal times for code inspections occur when code and unit tests are complete, and when the first integration tests are complete. Identify code inspections as milestones in the Project Plan.

Reference/Resource

Structured Walkthrough Process Guide

7.4 Conduct Unit and Module Testing

Responsibility

Development Team and Application Developer's Team

Applicability

Software can be developed faster and better with the assistance of efficient unit and module testing. Conducting the unit and module testing during the development phase could save time, identify errors earlier during the software construction evolution, and reduce costs down stream.

Description

Unit Testing: Unit testing mainly verifies if the units/classes function properly and meet their specifications. Unit testing is used to verify the input and output and proper operation for methods/class/objects. Successful testing indicates the validity of the methods/classes with traceability to the design specifications. During unit testing, test each method/class individually and verify the object/component interface for consistency with the design specification. Hence, each developer designs a unit level test and/or test case that invokes and exercises the methods/class validating proper execution, values, and expected outputs.

Module Testing: As specified above, the system is first divided in units which are developed and tested for their functionalities. These units are linked together into modules or subsystem components which are all assembled later to form a complete system. Hence, the module testing is to determine the proper operation of all subsystems and external subsystem interfaces, and execution paths and ensure they all work as the architect expected. Module testing is used to verify the input and output for each module. Successful testing indicates the validity of the function or sub-function performed by the module and shows traceability to the design. During Module testing, test each module individually and verify the module interface for consistency with the design specification. Test the important processing paths through the module for expected results. In addition, test the error handling paths. The tests should be written and conducted by the senior/lead developers of the development team. The end results of this test level should determine if the code is ready to fully assemble and migrate into the Systems Integration environment.

Both Unit and Module testing are driven by test cases and test data designed to verify the software requirements, and to exercise all program functions, edits, in-bound and out-of-bound values, and error conditions identified in the program specifications. If timing is an important characteristic of the unit and/or module, generate tests to measure time critical paths for average and worst-case situations.

Plan and document the inputs and expected outputs for all test cases in advance of the tests. Log all test results. Analyze, correct, and retest errors using the scenarios defined in the test cases. Repeat this cycle until all errors have been corrected. While unit testing is generally considered the responsibility of the developer, the project management team must be made aware of the unit test results.

Outputs

Completion of unit and module testing for a software component signifies internal project delivery of a component for integration with other components. Place all components having completed unit and module testing under configuration control as described in the IRM Standards/Business and Technical Standards/Operations Support Domain/Configuration and Change Management. These components form the allocated baseline. Configuration controls restrict changes to tested and approved software in the allocated baseline. Subsequent changes or additions to the software agreed upon in a Critical Design Review receiving stakeholder concurrence, supersede the former baseline and establish a new Product Baseline.

As needed, review and update the draft versions of the Integration and System Test Plans developed during the Detailed System Design (DSD) Phase. Updates are to reflect any changes made to the software design. Deliver the final versions of the Integration and System Test Plans to the project sponsor and user for review and approval. Maintain the approved plans in the Project Folder.

Maintain all test materials, including unit test inputs, outputs, results and error logs in the Project Folder.

Review Process

Conduct peer reviews of the test materials. Perform structured walkthroughs on any updated plans.

Template

Unit Test Scenario Checklist

Reference/Resource

Structured Walkthrough Process Guide

7.5 Establish Allocated Baseline

Responsibility

Development Team and Application Developer's Team

Applicability

To ensure all projects are fully unit tested.

Description

An allocated baseline is an approved "build" of the software. After the first build has been completed, tested, and approved by the project management team or the lead application developer, the initial allocated baseline is established. Obtain approval for subsequent versions of an allocated baseline. The approved allocated baseline for one build supersedes its predecessor.

Conduct internal tests consistently, including:

- **Regression tests** - to verify the capabilities of earlier builds continue to perform within subsequent builds. Regression tests can be performed either manually or automatically. The scope is similar to a functional test which allows a consistent, repeatable validation of each new release of a product.
- **Functional tests** - to verify where the build meets the functional and data requirements and to correctly generate each expected display and report. The process includes a series of tests using a wide range of normal and erroneous input data. Such tests can be validating the product's user interface, database management, ADA and security compliance, installation, networking, etc.
- **Performance and reliability tests** - to identify the performance and reliability thresholds of each build. This test involves an automated test package through simulation of a variety of normal, peak, and exceptional load conditions.
- **Software Security Vulnerability tests -** to identify software vulnerabilities in custom software code, interfaces, and databases. This test involves automated security vulnerability test tools.

Once the initial allocated baseline is established, any change to the baseline must be managed under the change control procedures described in the IRM Standards/Business and Technical Standards/Operations Support Domain/Configuration and Change Management. Approved changes to an allocated baseline must be incorporated into the next build of the software and revisions made to the affected outputs [e.g., Requirements Definition Document (RDD), Detailed System Design (DSD) Document, and Program Specifications].

Outputs

Document and maintain the test plan and associated internal build test procedures and results in the Project Folder. Identify any errors along with a description of the corrective action to be taken. Maintain the configuration control logs and records in the Project Folder.

Expand the Requirements Traceability Matrix developed in the Requirements Definition Phase. All outputs developed during the coding, unit testing, and building processes must be traced back to the project requirements and the DSD. This traceability ensures the software satisfies all the requirements and remains within the project scope. Maintain the expanded Requirements Traceability Matrix in the Project Folder.

Review Process

When the project management team deems necessary, conduct peer reviews, addressing the internal build test materials maintained in the Project Folder. As needed, conduct structured walkthroughs of any updated documents.

Reference/Resource

Structured Walkthrough Process Guide

7.6 Manage Requirements

Responsibility

Project Management Team

Applicability

Ensure sufficient information on the documentation as well as on monitoring the status report.

Description

Review the components developed and cross-reference with those in the Requirements Traceability Matrix.

Outputs

Each requirement identified in the Requirements Definition Document (RDD) must be traceable to one or more design entities. The traceability ensures the software satisfies the requirements and does not include inappropriate or extraneous functionality. Expand the Requirements Traceability Matrix developed in the Requirements Definition Phase to relate the General System Design (GSD) to the requirements. Maintain the expanded matrix in the Project Folder.

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix to ensure the required functionality is included.

Reference/Resource

Structured Walkthrough Process Guide

7.7 Develop Test Scenarios

Responsibility

Project Management Team

Applicability

Developing the test scenarios is an essential activity to validate the success functionality of a product. Usually, various scenarios are created based on the functional testing document. This will expose the functional specification document to critical interpretation from an entirely different perspective. The Project Management Team and Application Developer's Team will have a chance to re-evaluate the objective of the specifications.

Description

As with many terms, scenario means different things to different people. Test scenarios are used to control the test cases to be processed and how to process them. The file may contain one or more scenarios and be cross-referenced. Test scenarios are based on the requirements, the General System Design (GSD), and Detailed System Design (DSD) Documents. Create the scenarios to identify likely production situations. Document the test scenarios in the Test Scenarios Template or recorded within software test tools. However the scenarios are recorded, essential information to include are:

- Version
- Build number
- Retry
- Test Scenario Number
- Process Number
- Environment
- Test scenario description
- Objective
- Assumptions and constraints
- Test files/test data

Outputs

Complete a Test Scenarios for each process and/or requirement to be tested. Maintain the Test Scenarios in the Project Folder.

Review Process

Conduct reviews for the Test Scenarios to ensure the scenarios tested match those detailed in the GSD and DSD Documents.

Template

Test Scenarios Template

Reference/Resource

IRM Standards/Business and Technical Standards/Application Domain/Testing

7.8 Plan Transition to Operational Status

Responsibility

Project Management Team

Applicability

A transition plan is needed to ensure a smooth transition from development to implementation and to the operational status.

Description

A successful transition from acceptance testing to full operational use depends on planning long before the software is installed in its operational environment. During planning, operational needs associated with the software are quantified. Procedures to be used in performing the transition are described. Rely on experience and data gathered from previous, similar projects to define the needs.

Outputs

Develop a transition plan, describing the detailed plans, procedures, and schedules guiding the transition process. Coordinate the development of the plan with the operational and maintenance personnel. Items to be considered when preparing the transition plan include:

- Detailed operational scenarios, describing the functions to be performed by the operational staff, support staff, maintenance staff, and users.
- The number, the qualifications, the training requirements of personnel, and an indication when they must be in place.
- A documented release process. If development is incremental, a definition of the process, schedule, and acceptance criteria for each release.
- A description of data generation or migration, including the transfer or reconstruction of historic data. Allow ample time for the project sponsor and user to review the content of reconstructed or migrated data files. (This helps reduce the chance of errors.)
- Problem identification and resolution procedures for the operational software.
- Configuration management procedures to be used for the operational product. Ideally, configuration methods used during development can continue to be used for the operational product.
- Delineation of the scope and nature of support provided by the project management team during the transition period.
- Identification of the organizations and individuals responsible for each transition activity, ensuring responsibility for the software by the operations and maintenance personnel increases progressively.

- Identification of products and support services needed for day-to-day operations, or enhance operational effectiveness.

Review Process

Conduct a structured walkthrough to ensure the transition plan is logical, accurate, and complete. Involve the operational and maintenance personnel.

Template

Transition Plan

Reference/Resource

Structured Walkthrough Process Guide

7.9 Generate Operating Documentation

Responsibility

Project Management Team /Technical Writer

Applicability

Various applications are suitable to generate the operating documentation. One is the Web-based front ends. For a smaller scale project, hardcopy documentation might be sufficient. Regardless of the format, care and planning is needed to produce helpful, coherent information.

Description

The Department of Human Services (DHS) standard for operating documentation pertains to managing, maintaining, and supporting the infrastructure resources of DHS. It outlines the guidelines and tools used at the operational level of the infrastructure. The standards allow DHS to provide clients with the greatest level of availability to resources, resulting in improved services. The infrastructure resources include the servers, routers, other agency assets, databases, applications, networks, and Internet components necessary to conduct the automated business functions of DHS.

The Operating Documentation consists of the Service Level Objective (SLO) for Gathering Metrics and Application Health Check.

Procedure

Some of the issues to consider when obtaining electronic documentation include:

- **Compatibility** - all components are to be compatible, provide for standard configurations, and be managed centrally allowing for expeditious problem resolution.
- **Simpler configuration** - since components need to be interoperable, the configurations tend to be less complex. This may lead to selecting a smaller pre-qualified group of vendors to supply the components.
- **Use of Commercial-Off-The-Shelf (COTS)** - decreases compatibility issues, provides a larger customer base for support, and allows for concentration on DHS's business requirements rather than software installation and maintenance.
- **Use reputable vendors** - third party products from reputable vendors provide a greater level of stability to DHS. In addition, the vendors tend to conform to industry standards to further improve that stability.
- **Future growth** - the components need to provide for DHS's current and future demands.
- **Recoverability -** continued service to clients is a paramount issue needing a robust disaster recovery plan.

- **Total Cost of Ownership** - the components used will provide cost effective alternatives for ownership and replacement.
- **Project management tools** - effective methods for identifying, tracking, and resolving problems arising in a production environment are necessary.
- **Transparent technology** - the framework components do not hinder the definition and resolution of problems arising in DHS. In addition, the technology provides for maximum scalability and allows for a wider range of configurations.
- **Enterprise management tools** - provide adequate metrics and reports to support the management of the assets of DHS. Some areas monitored include system capacity, availability, and stability.
- **Problem routing** - the components having appropriate alert mechanisms route the definitions of problems to the appropriate resources.
- **Remote access and management** - provides the ability to remotely access and manage the assets of a particular system within DHS.
- **Limited customer view** - the customer focus is only on their area of responsibility.
- **Asset management** - the maintenance of information on DHS assets must be current. Tools help with the inventory activities.

Outputs

Complete the documentation for the SLO for gathering metrics and Application Health Check.

Review Process

Conduct a structured walkthrough to ensure the operating documentation is logical, accurate, and complete. Involve the operational and maintenance personnel.

Templates

Service Level Objective (SLO) for Gathering Metrics

Application Health Check

Reference/Resource

Structured Walkthrough Process Guide

7.10 Refine Training Plan

Responsibility

User Education Team

Applicability

Improve and perfecting the Training Plan.

Description

The training plan outlines the training needed to successfully implement and operate the software, and addresses the training to be provided to the project sponsor, users, operational, and maintenance staff.

Training must address the knowledge and the skills required to use the system effectively. Training accomplishes:

- Provide trainees with the specific knowledge and skills necessary to perform the work.
- Prepare complete training materials to describe the software and instruct the trainees. Training should leave the trainees with the enthusiasm and desire to use the new product.
- Account for the knowledge and skills the trainees bring with them, and use the information as a transition to learning new material.
- Anticipate the needs for follow-up training after the software is fully operational, including the possibility of refresher courses, advanced training, and for new personnel.
- Build in the capability to easily update the training as the software evolves.
- Address the need for extra training if new hardware or systems software is introduced.

The project sponsor and users are to be involved in the planning activities to determine the training needs for all categories of stakeholders (managers, users, operational, and maintenance staff).

Outputs

Issues addressed by the Training Plan include:

- Identification of personnel to be trained. A list of trainees is reviewed with the project sponsor and users to ensure all personnel to receive training are identified.
- A definition of the overall approach to training along with required courses and materials.

- Delineation of the scope of the training needed for managers, users, operational and maintenance staff.
- Explanation of how and when training is to be conducted, including instructor qualifications, learning objectives, and mastery or certification requirements (if applicable).
- Identification of any skill areas for certification, if required.
- Establishment of a preliminary schedule for the training courses. The schedule must reflect training requirements and constraints outside of the project. It may also include the identification of critical paths in the training schedule such as the period for the software installation and conversion to production status.
- The required courses are defined, including an outline of content and sequence.
- Reference to the organization's training policy for meeting training needs.
- Steps to ensure managers receive orientation on the training program.
- The training courses prepared at the organizational level are developed and maintained according to Commonwealth and organizational standards.
- A waiver procedure for training the individuals possessing the knowledge and skills required to perform their designated role.
- Methods used for evaluating the effectiveness of the training, such as questionnaires, surveys, and post training evaluations.
- Ensure training records are properly maintained.

Maintain the revised Training Plan in the Project Folder. Review and update the plan during the Software Integration and Testing Phase.

Review Process

Conduct a structured walkthrough to ensure the initial Training Plan is accurate and complete.

Reference/Resource

Structured Walkthrough Process Guide

7.11 Finalize Capacity Plan

Responsibility

Development Team and Project Team Analysts

Applicability

The capacity plan, if done correctly and proactively can help avoid downtime. The Capacity Plan should include prediction of when a given application might outgrow its existing server or when more users are added to accessing the application.

Description

The Capacity Plan estimates the capacity for the application by usage, network bandwidth, disk storage capacity, demographic profile environment, load test environment, production environment, and batch/FTP capacity. Continuously revise the plan throughout the project to ensure the latest capacity requirements are captured and verified.

Outputs

Design and document the finalized capacity plan in accordance with project design standards. Maintain the final Capacity Plan in the Project Folder.

Review Process

Conduct a structured walkthrough to verify the capacity requirements are correct and complete.

Reference/Resource

Structured Walkthrough Process Guide

7.12 Establish Integration Test Environment

Responsibility

Project Management Team

Applicability

Establish various test teams and ensure the test systems are ready.

Description

Based on the hardware and software specifications listed in the System Architecture document and the Acquisition and Installation Plan, the Integration and System Test Environment is setup and prepared. This step is necessary prior to the start of the testing phase. All requirements specified must be delivered and validated according to the specifications. Any changes to the original specifications are to be documented in the System Architecture document to ensure the new specifications are captured. Install and test the testing validation tools.

Outputs

All validated output shall be recorded and maintained in the Project Folder.

Review Process

Conduct structured walkthroughs of all the test systems in placed.

Reference/Resource

Structured Walkthrough Process Guide

7.13 Develop Batch Operations

Responsibility

Project Management Team

Applicability

Batch operations may be needed to operate the application efficiently and effortlessly.

Description

The Batch Operations Manual (BOM) is structured to give a step-by-step overview of batch operations. The user can either peruse the whole document top-down or focus on key subsections of interest. The BOM contains the batch operations for the application and all the relevant information to successfully operate the batch jobs when the application is released. Information captured includes:

- Batch Job description
- Pre-event
- Post-event
- Frequency
- Start Time
- Run Days
- Expected Run Time
- Input File
- Output File
- Escalation Process
- Contact Person

The document is used as reference information to assist the Department of Human Services (DHS) Batch Operations with detailed information on an application's batch strategy and approach.

Outputs

Complete and maintain the Batch Operations Manual in the Project Folder.

Review Process

Conduct a structured walkthrough to verify the Batch Operations are correct and complete.

Template

Batch Operations Manual

Reference/Resource

Structured Walkthrough Process Guide

Tasks

Along with completing the Batch Operations Manual, include the task, as needed:

7.13.1 Develop Batch Operations Services Request

7.13.1 Develop Batch Operations Services Request

Responsibility

Project Management Team

Applicability

A Batch Operations Service Request is more efficient than multiple calls made to a single operation.

Description

The Batch Operations Services Request captures the information needed to add, change, or delete a batch operation service request. The document is completed and submitted to the appropriate parties and reviewed to validate the need for the batch request and to see if it may adversely affect the system or other batch job schedules.

> *Note: (DHS Mainframe specific) – A BOPWK11 traveler is used for batch requests. Contact your supervisor for more information.*

Outputs

Complete the Batch Operations Services Request, as needed. Maintain the request in the Project Folder.

Review Process

Review the Batch Operations Services Request to ensure the information captured is accurate and complete.

Template

Batch Operations Service Request

7.14 Conduct Phase Review

7.14.1 Conduct Peer Reviews

Peer reviews are conducted within the project management team to review each of the Requirements Definition Phase deliverables. This is not a formal process but rather a review to ensure team members at the same level agree to the deliverables produced and to leverage the knowledge and experience of others.

7.14. 2 Conduct Structured Walkthroughs

Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of outputs and deliverables. The walkthroughs are scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage determines the frequency of the walkthroughs.

Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants. In some cases, it may be beneficial to include software users in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same. For further information, refer to the Structured Walkthrough Process Guide.

7.14.3 Conduct In-Phase Assessments (IPA)

If an IPA is required for this project, the work will be performed by an independent reviewer. The reviewer delivers the IPA report to the project management team which maintains the document in the Project Folder. For further information, refer to the In-Phase Assessment Process Guide.

7.14.4 Conduct Phase Exit

A Phase Exit is conducted at the end of each phase of development or maintenance. The process ends with the receipt of concurrence from the designated approvers to proceed to the next phase. Concurrence indicates all known issues have an acceptable plan for resolution. The project management team maintains the document in the Project Folder. For further information, refer to the Phase Exit Process Guide.

8.0 Software Integration Testing

Applicability

This phase is for all systems where multiple components are developed or revised or where one component is likely to affect the functionality of other components either directly or indirectly. This phase is relevant to nearly all systems project efforts.

Description

Software integration and testing activities focus on interfaces between the components of the software. Items tested for include functional correctness, software compatibility, system stability, overall system operability, security, and performance. Software integration and testing performed incrementally provides feedback on quality, errors, and design weaknesses early in the integration process.

In this phase, software components are integrated and tested to determine whether the software meets predetermined functionality, performance, quality, interface, and security requirements. Once the software is fully integrated, system testing is conducted to validate that the software will operate in its intended environment, satisfy all user requirements, and is supported with complete and accurate operating documentation. The systems integration testing is used to confirm that the entire system behaves as the user expects. Systems Integration tests should be derived from the user-level use cases which are designed to exercise and validate the code paths the users will execute. During this test phase software vulnerability testing as well as preliminary Load and Performance testing should be conducted. The end results of this test level should determine if the code is ready to migrate into the Pre-SAT and/or Systems Acceptance Testing (SAT) environment (reference Section 8.4)t.

Inputs

Inputs to this phase include:

- Requirements Traceability Matrix *(expanded)*
- Test Plan Document (Systems Integration Test strategies, test files, data sets, etc.)
- Allocated baselines (Application Code)
- Operating Documentation
- Training Plan

High-Level Activities

This section is divided into sub-sections focusing upon the high-level activities performed during this phase. The activities represent the minimum requirements for a large software development effort. Notes are provided to assist in customizing the lifecycle phase requirements to accommodate project scalability. The high-level activities for the Software and Integration Phase consist of:

8.1 Conduct Integration and System Testing

8.2 Conduct Load Testing

8.3 Conduct Regression Testing

8.4 Establish Acceptance Test Environment

8.5 Conduct User Acceptance Test

8.6 Revise Test Plan

8.7 Manage Requirements

8.8 Refine Operating Documentation

8.9 Finalize Training Plan

8.10 Conduct Phase Review

Outputs

The outputs listed are the minimum requirements for a large software project. The size and complexity of the project determines deviations in the content and delivery of the outputs. Explanations of the outputs are provided under the applicable activities described below:

- Test Report
- Requirements Traceability Matrix (final)
- Operating Documentation (*revised*)
- Training Plan (*final*)

A matrix showing the outputs associated with each high-level activity is provided in Exhibit 8.0-1, Software Integration and Testing High Level Activities and Outputs.

Review Process

Quality reviews are necessary during this phase to validate the product and associated outputs. The activities appropriate for quality reviews are identified in this section and section 2.3 Quality Reviews section. The time and resources needed to conduct the quality reviews are to be reflected in the project resources, schedule, and Work Plan Standard.

Reference/Resource

Section 2.3 Quality Reviews, provides an overview of project reviews.

10.0 Operational Support Phase explains the software maintenance process.

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide Checklist

Software Integration & Testing Phase Checklist

Exhibit 8.0-1 Software Integration and Testing High Level and Outputs

High Level Activities		Outputs	Deliverable
8.1	Conduct Integration and System Testing	Test Report (Integration and System Test)	Y
8.2	Conduct Load Testing	Test Report (Load Test)	Y
8.3	Conduct Regression Testing	Test Report (Regression Test)	Y
8.4	Establish Acceptance Test Environment	Documented attributes and constraints of the acceptance test environment	N
8.5	Conduct User Acceptance Test	Test Report (Acceptance Test)	Y
8.6	Manage Requirements	Requirements Traceability Matrix (*final*)	Y
8.7	Refine Operating Documentation	Operating Documentation (*revised*)	Y
8.8	Finalize Training Plan	Training Plan (*final*)	Y
8.9	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	N

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

8.1 Conduct Integration and System Testing

Responsibility

Project Management Team

Applicability

Integration and integrations testing is required whenever multiple modules are developed, particularly when multiple developers or development teams are involved. It is also required when a single component is developed that can adversely impact the performance of the entire system or any part of it and the interface must be proven to comply with specifications and requirements. System testing is always required except where the project is small enough that unit testing covers the system test requirements.

Description

Integration and Integration Testing

During software integration, the software components developed by the development team, along with off-the-shelf software, and reusable code or modules obtained from other sources are assembled into one integrated product. Each software module and sub systems component is tested in a systematic manner in accordance with the Integration Test strategies and specifications are outlined in the overall systems test plan document. An incremental approach to integration enables verification that the product is working properly as additional components are integrated. It also allows for easier problem isolation and resolution.

System Integration Testing (SIT) is used to confirm that the entire integrated system and associated components and interfaces behave as the end user expects and in compliance with systems technical and performance specifications. Systems Integration tests should be derived primarily from the user-level use cases which are designed to exercise and validate the code paths the users will execute. During this test phase software vulnerability testing as well as Load and Performance testing should be conducted. The end results of this test level should determine if the code is ready to migrate into the Pre-SAT and/or Systems Acceptance Testing (SAT).

Integration testing is a formal procedure carefully planned and coordinated with the completion dates of the unit-tested modules. Integration testing begins with a software structure where called sub-elements are simulated by stubs. A stub is a simplified program or dummy module designed to provide one or more responses provided by the real sub-element. A stub allows testing of calling program control and interface correctness. As testing proceeds, unit-tested modules replace the stubs. The process continues until the entire system has been integrated and then fully tested in the systems integration environment.

Integration testing may be performed using “bottom-up” or “top-down” techniques. Most integration test plans make use of a combination of both bottom-up and top-down techniques. Scheduling constraints, and the need for parallel testing, may affect the testing approach.

Bottom-up

This approach incorporates one or more modules into a build; tests the build; and then integrates the build into the software structure. The build is normally comprised of a set of modules performing a major function of the new system. Initially, the function may be represented by a stub later replaced when the build is integrated.

Top-down

In this approach, individual stubs are replaced so the top-level control is tested first, followed by stub replacements moving downward in the software structure. All modules comprising a major function are integrated thereby allowing an operational function to be demonstrated before completion of the entire system.

System Testing

During system testing, the complete software is tested to ensure the product meets all requirements. System response timing, memory, performance, security, and the functional accuracy of logic and numerical calculations are verified under normal and high-load conditions. Query and report capabilities are exercised and validated. All operating documentation is verified for completeness and accuracy.

System testing is conducted on the system test bed, using the test cases and methodology described in the System Test Plan. The system test bed is to be as close as possible to the actual production system. The results of each test are recorded, and upon completion, included as part of the project test documentation.

When errors are discovered, the test team leader determines the severity of the error and the necessary subsequent action. If appropriate, minor problems can be corrected and regression tested by the development team within the time allotted for the system test.

Major problems may cause the suspension of system testing. System testing must be rescheduled to begin after all the problems are resolved. Regardless of the severity, all corrections or changes to the software must be controlled under the established configuration management mechanisms.

In some projects, user representatives are involved in the system test. In these cases, it is important to brief the participants concerning their role, responsibilities, and expectations. The briefing includes procedures for error correction, configuration management, and re-testing.

Outputs

Generate a final Integration Test Report at the completion of integration testing, specifying any unresolved difficulties requiring management attention. Maintain the Test Report in the Project Folder.

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix and the final Test Report.

Template

Integration System Test Report

Reference/Resource

Structured Walkthrough Process Guide

8.2 Conduct Load Testing

Responsibility

Project Management Team or Independent Test Team

Applicability

Whenever system changes or a new process is considered at risk in terms of process time constraints or data volume a system load test should be performed.

Description

Load testing can use scripts, batch files and user testers to place normal and exceptional user loads on the system infrastructure simulating production conditions. Load testing should test likely user loads on a normal day as well as exceptional loads. Exceptional loads might include the data volume after a holiday weekend or the anticipated maximum load that is anticipated over a two year period. The number of and types of users as well as data volumes are estimated by the development team.

The information tracked in the load test report includes the processes tested, data volume tested, data scenarios tested, the number of users simulated, and the types of user behaviors being simulated. Track the ramp up time, total load test time, and throttling occurring for each load test.

All discrepancies must be resolved before the software may be installed.

Outputs

The load test report becomes part of the overall test report. It documents the system testing and includes any discrepancies with specifications and the requirements.

Review Process

Conduct a structured walkthrough of the test report.

Reference/Resource

Structured Walkthrough Process Guide

8.3 Conduct Regression Testing

Responsibility

Project Management Team or Independent Test Team

Applicability

Regression testing should always apply whenever any software change is likely to impact the overall operation of the system or subsystem. The more complex the software; the more imperative regression testing becomes. When a user's business process is at risk, regression testing is necessary.

Description

Regression testing (i.e. re-runs of previous tests) verifies when code changes have not adversely impacted existing functionality. It confirms no additional defects are introduced when repairing defects or adding new functionality to 'approved' code.

Manual regression testing must be completed by the responsible party in each test phase. Automated regression testing may be completed by a designated team.

Regression testing occurs in every test phase. When code changes are made, the logic must be retested in the unit test, the function/integration test, the system test, etc., until the fixed logic reaches the area where the defect occurred. Automated regression testing will typically be employed when the system test phase has been completed.

The decision to perform regression testing must be based on the risk introduced by the change, the size or impact of the change, or the criticality of the business functions impacted. Only those cycles affected by problems need to be re-tested.

When applicable the regression test plan should be revised for future use, so that subsequent releases of the software will be retested to account for the current revisions, updates or corrections. When a regression testing tool provides for saving of tests, these tests might need to be revised.

It is advisable to use a software product that can automate the testing process and that can save the battery of tests. This is recommended so as to eliminate human error, simplify management of testing and to reduce the labor intensive nature of regression testing.

Outputs

The regression test report is included within the overall Integration System Test Report. Maintain the Integration System Test Report in the Project Folder.

Review Process

Conduct a structured walkthrough of the Integrated System Test Report.

Reference/Resource

Structured Walkthrough Process Guide

IRM Standards/Business and Technical Standards/Application Domain

ITB-APP001 Business Solutions Center of Excellence (BSCoE)

8.4 Establish Systems Acceptance Test Environment(s)

Responsibility

Project Management Team and Project Team Specialists

Applicability

Complete whenever the acceptance test environment is not already provided. It may also be required when the test plan makes necessary accommodations for the acceptance test that the current environments do not adequately provide.

Description

Systems Acceptance Test (SAT): Based on the hardware and software specifications listed in the System Architecture document and the Acquisition and Installation Plan, the Acceptance Test Environment must be setup and prepared. Typically the acceptance test environment should be as much like the production environment as possible. This step is necessary prior to the start of the testing phase. Differences between the production environment and the acceptance test environment should be documented and presented to the system sponsor's and user test teams. All the requirements specified must be delivered and validated according to the specifications. Any changes to the original specifications need to be documented in the System Architecture document in the appropriate section to ensure the new specifications are captured.

Pre-Systems Acceptance Test: Pre-Systems Acceptance test (Pre-SAT) is not a separate test environment but is a tactical and/or procedural step in the software test phase that provides the software development team exclusive access to the SAT environment for a predefined time period prior to beginning end-user acceptance testing. This step may be necessary prior to the start of the end-user acceptance testing phase due to known constraints within the Systems Integration Test Environments that may limit thorough end-to-end testing of all systems components and/or interfaces to other internal or external systems. The project and test plans should be aligned for software development initiatives that will require a Pre-SAT phase. Differences between the systems integration environment and the acceptance test environment should be formally documented in the test plan document as well as any Pre-SAT systems tests that will be conducted in the SAT environment and presented to BIS technical managers.

Outputs

Documented attributes and constraints of the acceptance test environment.

Maintain the updated systems architecture document in the project folder.

Revised Test Plan Document

8.5 Conduct Pre Systems Acceptance and User Acceptance Tests

Responsibility

Acceptance Test Team

Applicability

Except where output from other system testing is considered adequate by the development team, users, project sponsor and application owner for determining acceptance, user acceptance testing is applicable.

Description

Pre-Systems Acceptance Test (Pre-SAT): Pre-SAT environment(s) is a functional staging environment within the SAT environment used by the development teams to: 1) Conduct final Systems Integration tests and/or additional tests unable to be performed in SIT environment(s) due to the inherent constraints in the systems integration environment(s), 2) Ensure proper connectivity and proper operation of all interfaces and systems components, 3) Configure systems parameters in preparation for proper SAT execution.

User Acceptance Testing: Acceptance of delivered software is the primary objective of a software development project. Acceptance testing is used to demonstrate the software's compliance with the project sponsor's requirements and acceptance criteria.

Acceptance testing may be performed by the project management team, the project sponsor and users with support from the project management team, or by an independent verification and validation team. Whenever possible, users participate in acceptance testing to ensure the software meets all established requirements.

User or Systems Acceptance Testing (SAT) is the final testing phase that involves the end-users testing and validating of the production ready code. During test execution, the user actions are both structured and randomized to add to the realism of the testing and to better assess the reliability and response of the system. SAT is meant to validate compliance of both functional and non-functional requirements, systems proper operations, integration, and interoperability, and certify operational readiness to support live operational environments. The end results of this test level should determine if the code is ready to migrate into the Test for Production (TFP) environment (reference TPF outlined in Section 9.0 of this document).

Testing is complete when all tests have been executed correctly. If one or more tests fail, the problems are documented, corrected, and retested. If the failure is significant, the acceptance test process may be postponed until the problem is corrected.

With the involvement of new users in the process, problems undetected can be discovered. When this happens, not only might corrections or enhancements to the system become necessary, but also to the test plan itself.

The acceptance test report will summarize the test procedures executed, any problems detected and corrected, and the projected schedule for correcting any open problem reports.

Signoff by the project sponsor is generally their approval to proceed.

Outputs

The formal acceptance test report. Maintain this report as part of the Test Report in the Project Folder, Final Test Plan Document in the project folder.

Review Process

If required, perform an Operational Readiness Review or Go/No-Go Assessment upon completion of acceptance testing. This review is a combined quality assurance and configuration management activity focusing on the results of the acceptance test and the readiness of the software going into production. Include in the review:

- The functional configuration audit to determine whether the test records demonstrate if the product meets the technical requirements
- The physical configuration audit to determine whether the technical documentation is complete

During the Operational Readiness Review, examine the acceptance test results with the project sponsor and user. Document any problems along with solutions and action plans. Once the problems are resolved, the software is ready for formal acceptance by the project sponsor.

A successful Operational Readiness Review establishes the operational baseline (i.e., the final baseline) for the software. A review consists of:

- The software and the technical documentation describing the operational software
- The current functional baseline
- The product baselines for the configuration items comprising the system
- Other system-level technical documentation generated during the lifecycle

Each new release must be preceded by an Operational Readiness Review if the operational product requires enhancements or changes to correct problems. Establish the updated system documentation as a new operational baseline.

8.6 Manage Requirements

Responsibility

Project Management Team

Applicability

Formal requirements management during testing is necessary for all complex projects where development and testing might have actually or is perceived to have expanded outside of the requirements or otherwise has exceeded or strayed from the defined scope of the systems project.

Description

Review the entire test plan (script) and test result items and cross-reference with the requirements using the Requirements Traceability Matrix.

Perform this by adding all test plan (script) and test result item ids to the Requirements Traceability Matrix that was started during the requirements definition phase. Each requirement identified in the test plan and the test results document must be traceable to one or more design entity. The traceability ensures the software will satisfy the requirements and will not include inappropriate or extraneous functionality.

Outputs

The Expanded Requirements Traceability Matrix created during the Requirements Definition Phase is the primary product of this step. Maintain the expanded matrix in the Project Folder.

Review Process

Conduct a structured walkthrough of the Requirements Traceability Matrix to ensure all required functionality is included.

Reference/Resource

Structured Walkthrough Process Guide

8.7 Refine Operating Documentation

Responsibility

Project Management Team

Applicability

Complete whenever the process warrants instructions for operations or other support staff to manage, maintain, and support it.

Description

The Department of Human Services (DHS) standard for operating documentation provides standard guidelines and an organized method for providing guidelines and instructions for managing, maintaining, and supporting the infrastructure resources of DHS. It also outlines the guidelines and tools used at the operational level of the infrastructure. The standards allow DHS to provide its clients the greatest level of availability to those resources, resulting in improved service. The infrastructure resources include the servers, routers, other agency assets, databases, applications, networks, and Internet components necessary to conduct the automated business functions of DHS.

Revise the Operating documentation started in the Development Phase to make the instructions fully complete and accurate. Add any additional needs and revisions to the document based upon the Integration and System Testing Phase.

Outputs

Maintain the revised Operating Documentation in the Project Folder.

Review Process

Conduct structured walkthroughs for the Operating Documentation or set of user documents to ensure the documentation is complete, easy to use, and accurately reflects the software and functionality.

Reference/Resource

Structured Walkthrough Process Guide

8.8 Finalize Training Plan

Responsibility

User Education Team

Applicability

To the extent that the training plan is not finished, functionality of the application has changed from previous phases, the timeline has changed or that aspects of the system become more evident to the user education team where they were not before, the training plan may have to be updated.

Description

A Training Program defines the training needed to successfully implement and operate the software and addresses the training to be provided to the project sponsor, users, operational and maintenance staff.

The Training Plan is finalized during or immediately after the system testing is complete. This is done to ensure its readiness and usage for the Acceptance and Installation Phase.

Outputs

Maintain the Training Plan in the Project Folder

Review Process

Conduct a structured walkthrough to ensure the final Training Plan is accurate and complete. This is important to ensure that necessary changes in the content and the schedule are confirmed that otherwise might be missed.

Reference/Resource

Structured Walkthrough Process Guide

8.9 Conduct Phase Review

8.9.1 Conduct Peer Reviews

Peer reviews are conducted within the project management team to review each of the Requirements Definition Phase deliverables. This is not a formal process but rather a review to ensure team members at the same level agree to the deliverables produced and to leverage the knowledge and experience of others.

8.9.2 Conduct Structured Walkthroughs

Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of outputs and deliverables. The walkthroughs are scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage determines the frequency of the walkthroughs.

Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants. In some cases, it may be beneficial to include software users in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same. For further information, refer to the Structured Walkthrough Process Guide.

8.9.3 Conduct In-Phase Assessments (IPA)

If an IPA is required for this project, the work will be performed by an independent reviewer. The reviewer delivers the IPA report to the project management team which maintains the document in the Project Folder. For further information, refer to the In-Phase Assessment Process Guide.

8.9.4 Conduct Phase Exit

A Phase Exit is conducted at the end of each phase of development or maintenance. The process ends with the receipt of concurrence from the designated approvers to proceed to the next phase. Concurrence indicates all known issues have an acceptable plan for resolution. The project management team maintains the document in the Project Folder. For further information, refer to the Phase Exit Process Guide.

9.0 Acceptance and Installation

Applicability

This section is always applicable for all but the smallest projects. This is due to the fact that the Department of Human Services (DHS) procedures specify the formal acceptance process by the project sponsor must be completed before newly developed software is installed on a production system. Furthermore, the use of an installation plan is required for all but the most simple of system installations and installation must always follow Departmental procedures.

Description

Although many portions of the DHS methodology may be performed concurrently, some of the activities in the latter phases of the lifecycle are sequential. System testing must be complete before formal acceptance testing may begin. Likewise, the product must be formally accepted before installation can begin.

User acceptance activities for larger (and more critical) projects are more formal than those typically found in small and express projects. Regardless of the formality, the activities in this phase focus on product verification and acceptance by the project sponsor or designee with the subsequent installation activities.

The activities associated within this phase must be performed each time a major product release is implemented.

Inputs

Input to this phase includes:

- Operating Documentation
- Training Plan
- Conversion Plan

High-Level Activities

This phase is divided into high-level activities performed during the phase. These activities represent the minimum requirements for a large software development effort. Notes are provided to assist in customizing the lifecycle phase requirements to accommodate project scalability. The high level activities for the Acceptance and Installation Phase consist of:

9.1 Establish Test for Production Environment

9.2 Perform Test for Production Activities

9.3 Perform Installation Activities

9.4 Conduct Installation Tests

9.5 Establish Training Environment

9.6 Conduct User Training

9.7 Transition to Operational Status

9.8 Conduct Phase Review

Outputs

The size and complexity of the project determine the deviations in the content and delivery of the outputs. Explanations of the outputs are provided under the applicable activities. The minimum output requirements for a large software project include:

- Deployment Playbook
- Installation Test Materials
- Training Materials
- Operating Documentation (*final*)

A matrix showing the outputs associated with each high-level activity is provided in Exhibit 9.0-1, Acceptance and Installation High Level Activities and Outputs.

Review Process

Quality reviews are necessary during this phase to validate the product and associated outputs. The activities appropriate for quality reviews are identified in this section and section 2.3 Quality Reviews. The time and resources needed to conduct the quality reviews must be reflected in the project resources, schedule, and Work Plan Standard.

Reference/Resource

Section 2.3 Quality Reviews, provides an overview of the types of reviews to be conducted within a project.

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

Checklist

Acceptance & Installation Phase Checklist

Exhibit 9.0-1 Acceptance and Installation High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
9.1	Establish Test for Production Environment	Conform to System Architecture Review Board Document	Y
9.2	Perform Test for Production Activities	Conform to System Architecture Review Board Document	Y
9.3	Perform Installation Activities	Deployment Playbook	Y
9.4	Conduct Installation Tests	Installation Test Materials	Y
9.5	Establish Training Environment	Training Environment	Y
9.6	Conduct User Training	Training Materials	Y
9.7	Transition to Operational Status	Transition Plan (revised)	N
9.8	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	N

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

9.1 Establish Test for Production Environment

Responsibility

Project Management Team and Domain Area Specialists

Applicability

This step is applicable when a test for production (TFP) environment must be set up to validate and approve the application for the production environment. For many projects, particularly smaller ones where there is no new hardware or types of software, this step will not be necessary. This step is necessary if and when the Architecture Review Board (ARB) indicates so.

Description

Based on the hardware and software specifications listed in the System ARB document and the Acquisition and Installation Plan, the TFP Environment must be setup. This step is necessary prior to the start of the TFP activity. All the requirements specified must be delivered and validated according to the specifications. Any changes to the original specifications need to be documented in the System ARB document in the appropriate section to ensure the new specifications are captured.

Outputs

Conform to the System ARB document. Refer to 5.13 Develop Architecture Review Board (ARB) Document.

Review Process

Conduct structured walkthroughs to ensure test for production is properly established.

Reference/Resource

Structured Walkthrough Process Guide

9.2 Perform Test for Production Activities

Responsibility

Project Management Team

Applicability

This section applies to the extent that test for production (TFP) activities are deemed necessary over and above systems testing by any of the domain specialists, *e.g., database, network support*, or the project team.

Description

Perform the Test for Production prior to the installation activities. This activity is similar to the other installation activities, but, involves testing the final production environment, which is often of a different configuration than the systems used to develop and test the application. Therefore, time is spent to test the production environment to ensure the installation and configuration is met.

Any gap or discrepancies found in this phase will be recorded, evaluated and fixed prior to acceptance and installation of the final product. Within each cycle, operate test scripts in succession and produce copies of any printouts and screen dumps of any screens. Attach these to the test script. Compare these against the expected results and log any discrepancies as they arise on a formal test incident form.

Outputs

Conform to the System Architecture Review Board specification document.

Review Process

For any discrepancies found from the test, system test fixes shall be performed.

9.3 Perform Installation Activities

Responsibility

Project Management Team, Project Team, and Domain Area Specialists

Applicability

This section applies anytime installation is required and after all approvals have been obtained from the project sponsor and/or the system owner.

Description

The installation process involves loading, copying, or migrating, any required software and data to the production platform. Also included is the delivery of operating documentation and other support materials for each production site. The installation of firmware, hardware, and communications equipment may also be involved.

All installation activities need to be included in the "Deployment Playbook". This deliverable lists all the activities, roles, responsibilities, contacts, and duration times for the complete installation and deployment of the application.

If a current system exists, system and data conversions must be implemented in accordance with the procedures described in the Conversion Plan. Each data and file conversion includes a confirmation of data and file integrity. A comparison of data output from the current and the new system are compared and verified to be accurate.

At each installation site, the facility must be inspected to ensure the site preparation is complete and in accordance with the Installation Plan. Activities are initiated to complete any necessary preparations. An inventory must be conducted on all vendor-provided hardware, software, firmware, and communications equipment in accordance with the Acquisition Plan.

The procedure specified in the Installation Plan must be followed when installing the software, hardware, and other equipment. All installation activities, including those performed by vendors, must be monitored to ensure a correct and complete installation.

Procedure

The procedure (or a suitable substitute) to perform the installation activities includes:

- Coordinate the installation with the project sponsor, user representatives, operations staff, and other affected organizations.
- Ensure any necessary modifications to the physical installation environment are completed.
- Inventory and test the hardware supporting the software. This inventory must be performed in advance of the planned installation date to allow time for missing hardware to be obtained and malfunctioning equipment to be replaced or repaired.

- If the software requires an initial data load or data conversion, install and execute the tested programs to perform this process.
- Install the software onto the hardware platform.

Outputs

Maintain the Deployment Playbook in the Project Folder.

Review Process

Conduct structured walkthroughs to ensure the Deployment Playbook is accurate and complete.

Template

Deployment Playbook

Reference/Resource

Structured Walkthrough Process Guide

9.4 Conduct Installation Tests

Responsibility

Project Management Team

Applicability

This section applies whenever testing of the installed system or any of its components is required to ensure its operational worthiness.

Description

The integrity and quality of the installed software is verified by executing the installation tests defined in the Installation Plan. If the software being installed is a modification to an existing system, all remaining functions and code affected by the new software must be tested. An integration test plan is designed to validate all functions of the software product and to specify a standard set of test results and tolerances.

Any problems must be documented along with the appropriate corrective action. The use of a diagnostic package is recommended to identify problems quickly and allow for timely corrections. All hardware and software must be retested after a repair, replacement, or modification of hardware.

Each software component is certified upon successful completion of installation and checkout. When installation is complete, either a portion or all of the system test must be re-run using the acceptance test procedures to verify correct operation of the software.

Conduct installation testing to verify:

- Security functions
- Integration with the current software
- Interfaces with other systems
- System functionality based on the requirements

Outputs

Maintain the installation test materials in the Project Folder.

Review Process

Conduct structured walkthroughs of the installation test materials as appropriate.

Reference/Resource

Structured Walkthrough Process Guide

9.5 Establish Training Environment

Responsibility

Project Management Team

Applicable

Review this section when the systems architecture document and the training plan call for the establishment of a training environment.

Description

Based on the hardware and software specifications listed in the System Architecture document and the Acquisition and Installation Plan, the Training Environment must be setup and prepared. This step is necessary prior to the start of training. The requirements for training must be provided for within the training environment. Any changes to the original specifications need to be documented in the System Architecture document in the appropriate section to ensure the new specifications are captured.

Outputs

Training environment is set up for project manager, operation and maintenance staffs. The updated system architecture document should be restored to the project folder.

Review Process

Conduct structured walkthroughs of the training environment. Operation and maintenance manuals are complete and available for use during the training sessions.

Reference/Resource

Structured Walkthrough Process Guide

9.6 Conduct User Training

Responsibility

User Education Team, the Project Management Team, and any Project Team Support Staff

Applicability

This section applies if formal training is required. This will appear on the project schedule and a training plan will exist in the project folder given that this SDM is followed.

Description

User training is an important factor in the success of the product deployment. During training, most users will receive their first hands-on experience with the software. Operations and maintenance staff may also be trained to use, monitor, and maintain the product. The objective of the training is to provide the trainee with the basic skills needed to effectively use the software and raise confidence and satisfaction with the product.

The type of training will depend on the complexity of the software, along with the number and location of the users to be trained. Possible training formats include formal classroom training, one-on-one training, computer-based instruction, sophisticated help screens, and online documentation. Training is to be conducted according to the approved Training Plan.

Pilot testing of training materials needs to be considered. Participants may include project management team members, project sponsor, and user representatives. Those receiving training during pilot testing evaluate the content, instruction, and support materials. This feedback can be used to make improvements as appropriate.

If consecutive training classes are conducted, feedback is requested from the participants and used to continuously improve the training approach, methods, and materials.

At the completion of the training, users must have a thorough understanding of the input requirements of each transaction, the processing taking place, and the types of output generated.

Outputs

Submit the training materials to the project sponsor and user representatives for review and approval. Maintain the approved training materials in the Project Folder.

Training materials are subject to the same change control procedures as the other operating documents and need to remain current with changes to the software.

Review Process

Conduct structured walkthroughs of the training materials. The pilot test of the training session helps ensure delivery of a quality-training program.

Reference/Resource

Structured Walkthrough Process Guide

9.7 Transition to Operational Status

Responsibility

Project Management Team, the Project Team, the Maintenance Staff, and the User Education Team

Applicability

This step applies after all prior steps have been validated, reviewed and approved.

Description

After the project sponsor or user representative has formally accepted the product, the transition to operational status begins. The procedures identified in the Transition Plan are used to implement the transition process. Stress tests, or other operational tests, may be necessary to confirm a sound installation. The product may require checking to determine tolerances to adverse conditions, failure modes, recovery methods, and specification margins. Training and certification activities are completed, as needed. Any support to be provided by contracted services is checked to ensure the support starts on time.

The project team usually provides operational and technical support during the transition. Project team members who have a comprehensive understanding of the product need to be identified as potential support personnel. These individuals may be helpful in providing assistance in the areas of software installation and maintenance, test, and documentation of changes. Technical support may involve the analysis of problems in software components and operational procedures, the analysis of potential enhancements, and vendor supplied upgrades to software components.

Transition to full operational status is an event-oriented process not completed until all transition activities have been successfully performed. Support provided by project team personnel is typically withdrawn in a gradual sequence. This aids in the smooth operation of the product and supports user confidence in the product. A formal transfer of all responsibility to the maintenance staff is planned at the conclusion of the transition process.

Access to all Project Folder materials, operating documents, identification of any planned enhancements, and other pertinent records are turned over to the maintenance staff. The software access rules must be modified to provide access to the maintenance staff and to remove access from the project team and other temporary users. Programs, files, and other support software must be in the production library and deleted from the test library, where appropriate.

For major systems involving multiple organizations or interfaces with other systems, a formal announcement of the transition to production is recommended. The announcement must be distributed to all affected groups, along with names, telephone numbers, and e-mail addresses of the maintenance staff.

Outputs

Transition the system into operational status. Release the Project Folder materials, transition plan, operating documents, and other pertinent records to the maintenance staff.

Review Process

Complete all reviews related to the functionality before the system is placed into operational status.

9.8 Conduct Phase Review

9.8.1 Conduct Peer Reviews

Peer reviews are conducted within the project management team to review each of the Requirements Definition Phase deliverables. This is not a formal process but rather a review to ensure team members at the same level agree to the deliverables produced and to leverage the knowledge and experience of others.

9.8.2 Conduct Structured Walkthroughs

Structured walkthroughs are appropriate for reviewing the technical accuracy and completeness of outputs and deliverables. The walkthroughs are scheduled to review small, meaningful pieces of work. The progress made in each lifecycle stage determines the frequency of the walkthroughs.

Walkthroughs can be conducted in various formats, with various levels of formality, and with different types of participants. In some cases, it may be beneficial to include software users in walkthroughs. Management representatives do not participate in structured walkthroughs. Regardless of the variations in format and participants, the basic activity (peer review) and the major objectives (find errors and improve quality) of the structured walkthroughs remain the same. For further information, refer to the Structured Walkthrough Process Guide.

9.8.3 Conduct In-Phase Assessments (IPA)

If an IPA is required for this project, the work will be performed by an independent reviewer. The reviewer delivers the IPA report to the project management team which maintains the document in the Project Folder. For further information, refer to the In-Phase Assessment Process Guide.

9.8.4 Conduct Phase Exit

A Phase Exit is conducted at the end of each phase of development or maintenance. The process ends with the receipt of concurrence from the designated approvers to proceed to the next phase. Concurrence indicates all known issues have an acceptable plan for resolution. The project management team maintains the document in the Project Folder. For further information, refer to the Phase Exit Process Guide.

10.0 Operational Support

Applicability

More than half of the life cycle costs are attributed to the operations and maintenance of systems. In this phase, it is essential all areas of operations and maintenance are understood and addressed. The system is being used and scrutinized to ensure it meets the needs initially stated in the planning phase. Problems are detected and new needs arise. This may require modification to existing code, new code to be developed and/or hardware configuration changes. Providing user support is an ongoing activity. New users will require training and others will require training as well. The emphasis of this phase will be to ensure the users' needs are met and the system continues to perform as specified in the operational environment. Additionally, as operations and maintenance personnel monitor the current system they may become aware of better ways to improve the system and therefore make recommendations. Changes will be required to fix problems, possibly add features and make improvements to the system. This phase will remain in effect for as long as the system is in use.

Description

This section describes an iterative process for conducting operational support activities. The process includes a minimal set of criteria necessary for project management and quality assurance processes, control and management of the planning, execution, and documentation of software maintenance activities. The use of automated tools to facilitate requirements definition, design, coding, and system documentation is encouraged. The selection and implementation of tools may vary from one project to another.

The basic maintenance process model includes input, process, output, and control for software maintenance. It is based on the same software development principles and practices to lower risk and to improve quality during the planning and development phases of the lifecycle. The concept of the maintenance process model is based on planned software changes be grouped into scheduled releases and managed as projects. This proven approach allows the individuals who are given the maintenance task, the tools to plan, optimally use resources, leverage economies of scale, and control outcome in terms of both schedule and product quality.

Each organization performing software maintenance activities must have a local documented procedure for handling emergency changes not implemented as part of a scheduled release. Generally, the changes include fixes to correct defects and updates to meet unscheduled business or legal requirements. Emergency changes must be integrated into the next release for full regression testing and documentation updates.

Inputs

Batch Operations Manual (BOM) (final)

Operating Documentation (final)

Training Materials (final)

Phases

The activities to be performed during software maintenance are grouped into logically related segments of work called "phases." The phases are similar to those referenced in the planning and development phases of the software lifecycle. The phases consist of:

10.1 Problem/Modification Identification Phase

10.2 Analysis Phase

10.3 Design Phase

10.4 Development Phase

10.5 System Test Phase

10.6 Acceptance Phase

10.7 Delivery Phase

> *Note: The maintenance process model does not presume the use of any particular system development methodology (e.g., waterfall, or spiral). The process is valid regardless of size, complexity, criticality, software application, or software usage.*

The software maintenance phases can be tailored as appropriate. Phases and outputs may be combined to effectively manage the project. Concurrence by the designated approvers must be reached during the Analysis Phase regarding decisions to combine subsequent phases. An example of tailoring exists when an emergency production fix is required – in such a case, only Problem Identification (10.1), Analysis (10.2), Development (10.4) and Delivery Phase (10.7) might be completed.

Outputs

A matrix showing the outputs associated with each phase is provided in Exhibit 10.0-1 Operational Support Phases and Outputs.

Identify Systems Operations

Operations support is an integral part of the day to day operations of a system. The BOM is developed in previous SDLC phases. This documented the activities, tasks, and responsible parties and will need to be updated as changes occur. Systems operations activities and tasks need to be scheduled, on a recurring basis, to ensure the production environment is fully functional and is performing as specified. Systems operations key tasks and activities include:

- Ensuring the systems and networks are running and available during the defined hours of Operations
- Implementing non-emergency requests during scheduled outages, as prescribed in the BOM and automated, are documented in the operating procedures. These processes should comply with the system documentation

- Performing backups (day-to-day protection, contingency)
- Performing the physical security functions
- Ensuring contingency planning for disaster recovery is current and tested
- Ensuring users are trained on current processes and new processes
- Ensuring that service level objectives are kept accurate and are monitored
- Maintaining performance measurements, statistics, and system logs. Examples of performance measures include volume and frequency of data to be processed in each mode, order and type of operations
- Monitoring the performance statistics, reporting the results and escalating problems when they occur

Maintain Data / Software Administration

Data / Software Administration is needed to ensure the input data and output data and databases are correct and continually checked for accuracy and completeness. This includes ensuring that any regularly scheduled jobs are submitted and completed correctly. Software and databases are to be maintained at (or near) the current maintenance level. Data / Software Administration tasks and activities include:

- Performing a periodic Verification / Validation of data, correct data related problems;
- Performing production control and quality control functions (Job submission, checking and corrections);
- Interfacing with other functional areas for day-to-day checking / corrections;
- Installing, configuring, upgrading and maintaining database(s). This includes updating processes, data flows, and objects (usually shown in diagrams);
- Developing and performing data / database backup and recovery routines for data integrity and recoverability. Ensure documented properly in the BOM
- Developing and maintaining a performance plan for online process and databases
- Performing configuration/design audits to ensure software, system, parameter configuration are correct

Project Management

As much as possible, all software maintenance activity must be managed as a project in order to gain the benefits of project management and to enable tracking of activities and costs. The extent of project management activity will vary, and must be tailored according to the size, complexity, and impact of the change or enhancement.

Review Process

In each phase, conduct one or more structured walkthroughs to validate the outputs, In-Phase Assessments (IPA), and a Phase Exit as needed.

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

Checklist

Operational Support Phase Checklist

Exhibit 10.0-1 Operational Support Phases and Outputs

High Level Activities		Outputs	Deliverable
10.1	Problem/Modification Identification Phase	Validated Modification Requests	Y
10.2	Analysis Phase	Detailed Analysis Report Test Plan Project Plan	Y Y Y
10.3	Design Phase	Revised Modification List Updated design baseline Test Plan (revised) Detailed Analysis Report (revised) Verified Requirements Project Plan (*updated*) Documented Constraints and Risks	Y Y Y Y Y Y Y
10.4	Development Phase	Updated Design Documentation Updated User Documentation Updated Training Plan Statement of risk and impact to users Test Readiness Report	Y Y Y Y Y
10.5	System Test Phase	Test Report Test Readiness Report Project Plan (*updated*)	Y Y Y
10.6	Acceptance Phase	Product Baseline Functional Configuration Audit Report Project Plan (*updated*) Modification Request Log (*updated*)	Y Y Y Y
10.7	Delivery Phase	Physical Configuration Audit Report Version Description Document (VDD)	Y Y

10.1 Problem/Modification Identification Phase

Responsibility

Maintenance Team

Applicability

The Department of Human Services (DHS) Project Change Management process facilitates consistent change control techniques to ensure project scope management throughout the lifecycle of DHS Information Technology (IT) projects. It protects the integrity of currently approved Community of Practice (CoP) and associated system requirements, design documents, and baselined software functionality components.

The process is used to track changes identified after planning is finalized and the scope is baselined. Then it follows through baselines developed from the General System Design (GSD), Detailed System Design (DSD), and Development phases. By the time a project is nearing completion of the Development phase, changes to scope are to be discouraged until the product is implemented and maintenance changes arise.

Because the definition of project scope includes agreed-upon time, cost and resource parameters, a Project Change Request (PCR) must be generated for any project changes impacting the project's resources, cost, or duration. Bug fixes and incidents discovered during testing do not represent changes to scope and are therefore not covered by this procedure.

Description

In this phase, software changes are identified, categorized, and assigned an initial priority ranking. Each software change request is evaluated to determine its classification and handling priority. The classification types of maintenance are:

- **Corrective** – Change to the software after delivery to correct defects. The changes may be reported by the "Hot Lines" from the user community.
- **Adaptive** – Change to the software after delivery to keep it functioning properly in a changing environment. These changes may be initiated and tracked via the Data Processing Service Request (DPSR) system.
- **Emergency** – Unscheduled corrective maintenance required to keep a system operational.
- **Scheduled** – Change to the software after delivery on a routine basis to improve performance or maintainability. The changes may be initiated by the users and tracked by the DPSR system.
- **Perfective** – Change to the software after delivery to improve performance or maintainability. The changes may be initiated by the user and tracked by the DPSR system.

Any number of factors can drive the need for software modifications, including:

- Report of system malfunction
- Mandatory changes required by new or changed federal or state law
- New requirements to support business needs
- Minor enhancement or redesign to improve functionality or replace an obsolete system component
- Operational system upgrades and new versions of resident software.

These factors must be considered when assigning a priority to the modification request.

Inputs

Input to the Problem/Modification Identification Phase of software maintenance is one or more Modification Requests.

Process

If a modification to the software is required, activities to occur within the maintenance process include:

- Assign an identification number
- Categorize the type of maintenance
- Analyze the modification to determine whether to accept, reject or further evaluate
- Prioritize the modification according to:
  - Emergency (follow emergency change procedure and integrate into the next scheduled release or block of modifications)
  - Mandatory (e.g., legal, safety, payroll)
  - Required (has associated benefits; e.g., productivity gains, new business drivers)
  - Nice to have (lower priority)

Control

Identify and maintain the Modification Requests and process determinations in the Project Folder.

Outputs

Maintain the validated Modification Requests and the process determinations in the Project Folder. They include:

- Statement of the problem or new requirement
- Problem or requirement evaluation
- Categorization of the type of maintenance required
- Initial priority
- Verification data (for corrective modifications)
- Initial estimate of resources required

Review Process

Conduct one or more structured walkthroughs to validate the outputs.

Reference/Resource

Structured Walkthrough Process Guide

10.2 Analysis Phase

Responsibility

Project Management Team and Project Sponsor

Applicability

As needed, repeat the appropriate activities and tasks of the system development methodology (SDM).

Description

During the Analysis Phase, Modification Requests validated in the Problem/Modification Identification Phase, along with system documentation, are used to study the feasibility and scope of the modification. A preliminary Project Plan for design, test, and delivery is developed.

If documentation is unavailable or inadequate, and the source code is the only reliable representation of the software system, reverse engineering may be necessary to ensure the overall integrity of the system. Likewise, in cases where long-lived systems have overgrown the initial base system and are poorly documented, reverse engineering may be required.

For a smaller scope, or for local analysis on a unit level:

- Dissection of source code into formal units
- Semantic description of formal units and declaration of functional units
- Creation of input/output schematics of units

For a larger scope, or for global analysis on a system level:

- Declaration and semantic description of linear flows
- Declaration and semantic description of system applications (functions grouped)
- Creation of anatomy of the system (system architecture)

Modifications of a similar nature, (i.e., affecting the same programs) need to be grouped together whenever possible, and packaged into releases managed as projects. A release cycle must be established and published.

Inputs

Input to the Analysis Phase of software maintenance includes:

- Validate Modification Request
- Initial resource estimate and associated information
- Project and system documentation, if available

Process

Preliminary analysis activities include:

- Make a preliminary estimate of the modification size/magnitude
- Assess the impact of the modification
- Assign the Modification Request to a block of modifications scheduled for implementation
- Coordinate the modifications with other ongoing maintenance tasks

Once agreement has been reached regarding modifications, the analysis progresses and includes:

- Define firm requirements for the modification
- Identify elements of the modification
- Identify safety and security issues
- Devise a test and implementation strategy

When identifying the elements of the modification, all outputs affected must be examined. Identify each output including:

- A specification of the portion of the product to be modified
- The interfaces affected
- The user-noticeable changes expected
- The relative degree and kind of experience required to make changes
- The estimated time to complete the modification

The test strategy is based on input from the previous activity, which identifies the elements of modification. Define the requirements for at least three levels of testing: *unit tests*, *integration tests*, and *user-oriented functional tests*. In addition, identify the regression test requirements associated with the testing levels. Revalidate the test cases used to establish the test baseline.

Control

Control of the Analysis Phase activities includes:

- Retrieval of the relevant version of project and system documentation from the configuration control functions of the organization
- Review of the proposed changes and engineering analysis to assess technical and economic feasibility, and correctness
- Identification of safety and security issues
- Consideration of the integration of the proposed change within the existing software

- Verification that all appropriate analysis and project documentation is updated and properly controlled
- Verification the test functions of the organization are providing a strategy for testing the changes, and the schedule can support the proposed test strategy
- Review of resource estimates and schedules; verification of accuracy
- Technical review to select the problem reports and proposed enhancements to be implemented in the new release

Outputs

The output of the Analysis Phase includes:

- Detailed analysis report
- Updated requirements (including traceability list)
- Test strategy
- Project Plan

A project plan states how the design, implementation, testing, and delivery of the modification are to be accomplished with a minimal impact to current users.

Review Process

At the end of the Analysis Phase, perform a risk analysis. Using the output from the Analysis Phase, revise the preliminary resource estimate, and make a decision on whether to proceed to the Design Phase.

10.3 Design Phase

Responsibility

Project Management Team

Applicability

As needed, repeat the appropriate activities and tasks of the system development methodology (SDM).

Description

In the Design Phase, all current system and project documentation, existing software and databases, and the output of the Analysis Phase are used to design the modification to the system.

Input

Input to the Design Phase of software maintenance includes:

- Analysis Phase output, including:
  - Detailed analysis
  - Updated statement of requirements
  - Preliminary modification list (identification of affected elements)
  - Test strategy
  - Project Plan
- System and project documentation
- Existing source code, comments, and databases

Process

The process steps for the Design Phase include:

- Identify selected software modules
- Modify software module documentation (e.g., data and control flow diagrams, schematics)
- Create test cases for the new design, including safety and security issues
- Identify/create regression tests
- Identify documentation (system/user) update requirements
- Update modification list

- Document any known constraints influencing the design and identify risks. In addition, document planned or past actions mitigating risk.

Control

During the Design Phase of a modification, the control activities performed are:

- Conduct structured walkthrough(s) of the design
- Verify the new design/requirement is documented as an authorized change
- Verify the inclusion of new design material, including safety and security issues
- Verify the appropriate test documentation has been updated
- Complete the traceability of the requirements to the design

Outputs

The outputs of the Design Phase of software maintenance include:

- Revised modification list
- Updated design baseline
- Updated test plans
- Revised detailed analysis
- Verified requirements
- Updated Project Plan
- Documented constraints and risks

Review Process

Conduct structured walkthroughs, In-Phase Assessments (IPA), and a Phase Exit as needed.

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

10.4 Development Phase

Responsibility

Project Management Team

Applicability

As needed, repeat the appropriate activities and tasks of the system development methodology (SDM).

Description

The result of the Design Phase, current source code, and project and system documentation, is used to drive the programming effort.

Inputs

Input to the Development Phase of software maintenance includes:

- Results from the Design Phase
- Current source code, comments, and databases
- Project and system documentation

Process

The Development Phase tasks include:

Coding and Unit Testing

Implement the change into the code and perform unit testing. Other quality assurance and verification and validation processes may be required for safety related code.

Integration

The modified software is integrated with the system, and integration and regression tests are refined and performed, after the modifications are coded and unit tested, or at appropriate intervals during coding. All effects (e.g., functional, performance, usability, safety) of the modification on the existing system are assessed and noted. A return to the coding and unit testing tasks is made to remove any unacceptable impacts.

Risk Analysis and Review

Perform risk analysis and review periodically during the Development Phase rather than at the end, as in the Design and Analysis Phases. Use metrics/measurement data to quantify risk analysis.

Test Readiness Review

Conduct a Test Readiness Review to assess the team's preparedness to enter system testing. This is a self-assessment to determine if items including code, documentation, libraries, hardware, telecommunication lines, and schedules are ready for system test to begin on the scheduled date.

Control

Control of the Development Phase includes:

- Conduct structured walkthroughs of the code
- Ensure unit and integration testing are performed and documented in the Project Folder
- Ensure test documentation (e.g., test plans, test cases, and test procedures) are either updated or created
- Identify, document, and resolve any risks exposed during software and test readiness reviews
- Verify new software is placed under software configuration management control
- Verify training and technical documentation have been updated
- Verify the traceability of the design to the code

Outputs

The outputs of the Development Phase include:

- Updated software
- Updated design documentation
- Updated test documentation
- Updated user documentation
- Updated training materials
- Statement of risk and impact to users
- Test Readiness Review report

Review Process

Conduct structured walkthroughs, In-Phase Assessments (IPA), and a Phase Exit as needed.

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

10.5 System Test Phase

Responsibility

Project Management Team or Independent Testers

Applicability

As needed, repeat the appropriate activities and tasks of the system development methodology (SDM).

Description

System testing is performed on the modified system. Regression testing is a part of system testing performed to validate the modified code does not introduce new problems (i.e., not present in a prior release).

Inputs

Input to the System Test Phase of software maintenance includes:

- Test Readiness Review report
- Documentation, which includes:
  - System test plans
  - System test cases
  - System test procedures
  - User manuals
  - Design
- Updated system
- Updated Project Plan

Process

System tests are conducted on a fully integrated system and include:

- System functional test
- Interface testing
- Regression testing
- Test readiness review to assess preparedness for acceptance testing

Results of tests conducted before the test readiness review must not be used as part of the system test report to substantiate requirements at the system level. This is necessary to ensure the test organization does not consider these tests to replace the requirement for a complete “system test.”

Control

For maximum results, use an independent party to conduct system testing. The test function is responsible for reporting the status of the activities established in the test plan for satisfactory completion of system testing, before completing the system testing. Report the status to the appropriate reviewers before proceeding to acceptance testing.

Place the code listings, modification requests, and test documentation under Project Management Tools/Change Control in the Project Folder. The project sponsor is required to participate in the review to ensure the maintenance release is ready to begin acceptance testing.

Outputs

The outputs for the System Test Phase of software maintenance include:

- Tested and fully integrated system
- Test report
- Test Readiness Review report
- Updated Project Plan

Review Process

Conduct structured walkthroughs, In-Phase Assessments (IPA), and a Phase Exit as needed.

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

10.6 Acceptance Phase

Responsibility

Project Sponsor or designee

Applicability

As needed, repeat the appropriate activities and tasks of the system development methodology (SDM).

Description

Acceptance tests are conducted on a fully integrated system by the project sponsor, the user of the modification package, or a third party designated by the project sponsor. An acceptance test is conducted on the modified system, with software configuration management, in accordance with the application's IRM Standards/Business and Technical Standards/Operations Support Domain/Configuration and Change Management

Inputs

Input for the Acceptance Phase of software maintenance includes:

- Test Readiness Review report
- Fully integrated system
- User Acceptance Test Plan
- Acceptance test cases
- Acceptance test procedures

Process

The steps forming the process for acceptance testing are:

- Perform acceptance tests at the functional level
- Perform interoperability testing (to validate any input and output interfaces)
- Perform regression testing
- Conduct a Functional Configuration Audit (FCA)

The purpose of a FCA is to verify all specified requirements have been met. The audit compares the system's software elements with the requirements documented in the Requirements Definition Document (RDD), to ensure the modifications address all, and only, those requirements. The results of the FCA must be documented, identifying all discrepancies found, and the plans for the resolution.

Control

Control of acceptance tests includes:

- Results for the Functional Configuration Audit are reported to ensure all approved functionality is present in the system
- Confirmation is obtained from the change authority indicating the change has been successfully completed
- The new system baseline is established
- The acceptance test documentation is placed under software configuration management control

Outputs

The outputs of the Acceptance Phase include:

- New system baseline
- Functional Configuration Audit Report
- Acceptance Test Report
- Updated Project Plan
- Updated modification request log

Review Process

Conduct structured walkthroughs, In-Phase Assessments (IPA), and a Phase Exit as needed.

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

10.7 Delivery Phase

Responsibility

Project Management Team

Applicability

As needed, repeat the appropriate activities and tasks of the system development methodology (SDM).

Description

This phase describes the requirements for the delivery of a modified software system.

Inputs

Input to the software maintenance Delivery Phase is the fully tested version of the system as represented in the new baseline.

Process

The tasks for delivery of a modified system include:

- Notify the user community
- Develop an archival version of the system for backup
- Perform installation and training at the user facility

Control

Control for the Delivery Phase includes the following:

- Provide access to system materials for users, including replication and distribution
- Complete the version description document
- Place under software configuration management control

Outputs

The outputs of the Delivery Phase include:

- Physical Configuration Audit report
- Version Description Document (VDD)

The VDD contains information pertinent to the versioned release of the system being delivered. Information includes:

- System name
- Date delivered

- Version-number
- Release number
- Brief description of functionality delivered in the modification
- Prerequisite hardware and software with its associated version and release number

The current VDD is placed together with prior VDDs to form a complete chronology of the system from its initial implementation.

Review Process

Conduct structured walkthroughs, In-Phase Assessments (IPA), and a Phase Exit as needed.

Reference/Resource

Structured Walkthrough Process Guide

In-Phase Assessment Process Guide

Phase Exit Process Guide

Refresh Schedule

All guidelines and referenced documentation identified in this standard will be subject to review and possible revision annually or upon request by the DHS Information Technology Standards Team.

Guideline Revision Log

Change Date	Version	CR #	Change Description	Author and Organization
03/01/99	1.0		Initial creation	Original SDM team
09/01/05	2.0	.	Change to DPW Business and Technical Standards format. Updated the document to incorporate departmental procedures, federal regulations, and best practices. Moved from Application Domain to Business Domain	PMO-DCSS
05/01/07	2.1		Corrected links to the new OIT Portal Site	PMO-DCSS
06/15/07	2.2		Format Changes	PMO-DCSS
03/11/11	3.0		Incorporate current changes to SDM components	DEA
05/03/11	3.1		Incorporated Information Mgmt Deliverable Requirements	PMO-DEPPM
12/29/14	4.0		Change DPW to DHS, Corrected links, many corrections and format changes to improve readability.	PMO-DEPPM
03/29/16	4.0		Yearly Review	PMO-DEPPM

Exhibit 2.0-1 Software Lifecycle Phases and Deliverables

Feasibility Business Drivers Scope Definition High Level Requirements Business Case (Business and Technical perspectives) Executive Sponsorship Business Review Board Communities of Practice Project Initiation Charter and HLEs Initial Project Plan and scheduling All Phase Reviews	**Requirements Definition** Requirements Traceability Matrix (*initial*) Change Request Form Change Control Log Requirements Definition Document(RDD) Disaster Recovery Plan Description of Analysis Technique Test Plan (*initial, iterative revisions during build/construct and test phases*) Revise Detailed Project Plan and Schedule All Phase Reviews
General System Design Description of Design Technique General System Design (GSD) Document Requirements Traceability Matrix (*expanded)* System Architecture Document (*initial*) Capacity Plan (*initial*) Acquisition and Installation Plan Training Plan (*initial*) Conversion Plan (*initial*) Architecture Review Board Document All Phase Reviews	**Detailed System Design** Detailed System Design (DSD) Document Requirements Traceability Matrix (*expanded*) System Architecture Document (*revised*) Conversion Plan (*revised)* Electronic Commerce Security Assessment (ECSA) Document Test Plan (*revised)* All Phase Reviews
Software/Systems Development Application Code Unit Test Scenario Checklist Requirements Traceability Matrix (*expanded*) Unit/Module Test Plan Test Scenarios Operating Documentation Transition Plan (*initial*) Training Plan (*revised*) Capacity Plan (*final*) Batch Operations Manual Batch Operations Services Requests Test Plan (*final*) All Phase Reviews	**Software Integration and Testing** SIT Test Plan Test Report (System & Integration Test) Test Report (Load Test) Test Report (Regression Test) Test Report (Acceptance Test) Requirements Traceability Matrix (*final*) Operating Documentation (*revised)* Training Plan (*final*) All Phase Reviews
Acceptance and Installation SAT Test Plan Deployment Playbook Installation Test Materials Training Materials Operating Documentation (*final*) All Phase Reviews	**Operational Support Phase** Transition Plan (*revised*) All Phase Reviews

Note: A project may require deliverables in addition to those listed in this table.

Exhibit 2.2-1 Example of SDM Adapted for COTS Projects

COTS Phase	Project Phase	Deliverables
Business Feasibility Research	Project Initiation and Planning	Use DHS COTS Evaluation and Selection Guideline; COTS Project Plan (includes WBS)
Business Feasibility Research	Execution	COTS Project Plan (includes WBS)
Solicitation Phase	Execution	High Level Requirements
Preliminary Evaluation-Selection	Execution	Requirements Definition Document Systems Requirements Document Products to be Evaluated Architecture Review Board Document
Detailed Evaluation-Selection	Execution	Business and Technical Evaluation Matrix Architecture Review Board Document
Assessment-Recommendation	Execution	COTS Solution / Recommendation Draft COTS Executive Summary Document
Approval & Authorizations	Execution	Final COTS Executive Summary Document Architecture Review Board Document Sponsor Approval Letter
Procurements	Execution	Acquisition Plan
Implementation-Adoption		Transition Plan Implementation Playbook Operating Documentation Capacity Plan Conversion Plan Training Plan Systems Integration Test Plan & Report System Architecture Document Architecture Review Board Document Electronic Security Assessment Document
Testing-Validation	Requirements Definition	User Acceptance Test Plan Test Report
Life Cycle Management		LCM Governance Document

Exhibit 4.0-1 Requirements Definition High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
4.1	Manage Requirements	Requirements Traceability Matrix (*initial*) Project Change Request (PCR) Form	Y N
4.2	Select Requirements Analysis Technique	Description of Analysis Technique	N
4.3	Define Project Requirements	Requirements Definition Document (Process Model Narratives)	Y
4.4	Define Data Backup/Restore and Disaster Recovery Requirements	Data Backup/Restore Requirements Disaster Recovery Plan	Y Y
4.5	Define Data Requirements	Requirements Definition Document (Data Dictionary)	Y
4.6	Create Initial Logical Data Model	Requirements Definition Document (Logical Data Model)	Y
4.7	Compile and Document Project Requirements	Requirements Definition Document	Y
4.8	Develop Test Plan	Test Plan (*initial*)	Y
4.9	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is an output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

Exhibit 5.0-1 General System Design High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
5.1	Select Design Technique	Description of Design Technique	N
5.2	Determine Software Structure	Design Entities and Dependencies	Y
5.3	Develop General System Design	General System Design Document	Y
5.4	Design the User Interface	General System Design Document (User Interface)	Y
5.5	Create Data Model	General System Design Document (Data Dictionary)	
5.6	Create Logical Data Model	General System Design Document (Logical Data Model)	Y
5.7	Manage Requirements	Requirements Traceability Matrix (*expanded*)	Y
5.8	Design System Architecture	System Architecture Document (*initial*)	Y
5.9	Develop Capacity Plan	Capacity Plan (*initial*)	Y
5.10	Initiate Procurement of Hardware and Software	Acquisition & Installation Plans	N
5.11	Develop Training Plan	Training Plan (*initial*)	Y
5.12	Develop Conversion Plan	Conversion Plan (*initial*)	Y
5.13	Develop Architecture Review Board Document	Architecture Review Board Document	Y
5.14	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

Exhibit 6.0-1 Detailed System Design High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
6.1	Select System Architecture	System architecture recommendation	N
6.2	Develop Detailed System Design	Detailed System Design Document	Y
6.3	Refine Data Requirements	Detailed System Design Document (Data Dictionary)	Y
6.4	Design Physical Data Model	Detailed System Design Document (Physical Data Model)	Y
6.5	Manage Requirements	Requirements Traceability Matrix (*expanded*)	Y
6.6	Select Programming Standards	Programming Standards	N
6.7	Refine Test Plan	Test Plan (revised)	Y
6.8	Develop Proof of Concept/Prototype	Proof of Concept	N
6.9	Refine Conversion Plan	Conversion Plan (*revised*)	Y
6.10	Develop Electronic Commerce Security Assessment (ECSA)	Electronic Commerce Security Assessment (ECSA) Document	Y
6.11	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is an output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

Exhibit 7.0-1 Development High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
7.1	Develop Acquisition Plan	Acquisition Plan	N
7.2	Establish Programming Environment	Physical Installation of Hardware and Software	Y
7.3	Write Programs	Application Code	Y
7.4	Conduct Unit Testing	Unit Test Scenario Checklist Test Plan (*final*) Systems Test Plan	N Y Y
7.5	Establish Allocated Baseline	Internal build test procedures and results	Y
7.6	Manage Requirements	Requirements Traceability Matrix (*expanded*)	Y
7.7	Develop Test Plan	(Iterative plan aligned with software construction evolution, submitted as final article)	Y
7.8	Develop and Execute Test Scenarios	Test Scenarios (i.e., Unit, Module, Sys Integration, End-User Acceptance)	Y
7.9	Develop and Execute Test Scripts/Checklists/Results	(i.e., Load & Performance, Software Vulnerability, other ancillary tests and/or specific technology and infrastructure component testing)	Y
7.10	Test Document Repository	Test Document Repository (links all test plans, scenarios, scripts, and results per software release)	N
7.11	Plan Transition to Operational Status	Transition Plan	Y
7.12	Generate Operating Documentation	Operating Documentation	Y
7.13	Refine Training Plan	Training Plan (*revised*)	Y
7.14	Finalize Capacity Plan	Capacity Plan (*final*)	Y
7.15	Establish Integration Test Environment	Validated Output	Y
7.16	Develop Batch Operations	Batch Operations Manual Batch Operations Services Request	Y Y
7.17	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	Y

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

Exhibit 8.0-1 Software Integration and Testing High Level and Outputs

High Level Activities		Outputs	Deliverable
8.1	Conduct Integration and System Testing	Test Report (Integration and System Test)	Y
8.2	Conduct Load Testing	Test Report (Load Test)	Y
8.3	Conduct Regression Testing	Test Report (Regression Test)	Y
8.4	Establish Acceptance Test Environment	Documented attributes and constraints of the acceptance test environment	N
8.5	Conduct User Acceptance Test	Test Report (Acceptance Test)	Y
8.6	Manage Requirements	Requirements Traceability Matrix (*final*)	Y
8.7	Refine Operating Documentation	Operating Documentation (*revised*)	Y
8.8	Finalize Training Plan	Training Plan (*final*)	Y
8.9	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	N

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

Exhibit 9.0-1 Acceptance and Installation High Level Activities and Outputs

High Level Activities		Outputs	Deliverable
9.1	Establish Test for Production Environment	Conform to System Architecture Review Board Document	Y
9.2	Perform Test for Production Activities	Conform to System Architecture Review Board Document	Y
9.3	Perform Installation Activities	Deployment Playbook	Y
9.4	Conduct Installation Tests	Installation Test Materials	Y
9.5	Establish Training Environment	Training Environment	Y
9.6	Conduct User Training	Training Materials	Y
9.7	Transition to Operational Status	Transition Plan (revised)	N
9.8	Conduct Phase Review	Documentation from Peer Review, Structured Walkthrough, and In-Phase Assessment	N

A deliverable is output identified in the Project Plan and formally delivered to the project sponsor and other project stakeholders for review and approval.

Exhibit 10.0-1 Operational Support Phases and Outputs

High Level Activities		Outputs	Deliverable
10.1	Problem/Modification Identification Phase	Validated Modification Requests	Y
10.2	Analysis Phase	Detailed Analysis Report Test Plan Project Plan	Y Y Y
10.3	Design Phase	Revised Modification List Updated design baseline Test Plan (revised) Detailed Analysis Report (revised) Verified Requirements Project Plan (*updated*) Documented Constraints and Risks	Y Y Y Y Y Y Y
10.4	Development Phase	Updated Design Documentation Updated User Documentation Updated Training Plan Statement of risk and impact to users Test Readiness Report	Y Y Y Y Y
10.5	System Test Phase	Test Report Test Readiness Report Project Plan (*updated*)	Y Y Y
10.6	Acceptance Phase	Product Baseline Functional Configuration Audit Report Project Plan (*updated*) Modification Request Log (*updated*)	Y Y Y Y
10.7	Delivery Phase	Physical Configuration Audit Report Version Description Document (VDD)	Y Y