SECTION THREE

III

System Development Lifecycle

SECTION III: SYSTEM DEVELOPMENT LIFECYCLE
TABLE OF CONTENTS

Section III Introduction

There are currently many different methodologies employed for system development projects within New York State agencies. Many methodologies are driven by the application development tools, by the software architecture within which the application will operate, or by the "build versus buy" decision. There are standard phases and processes, however, that all system development projects should follow, regardless of environment and tools. This section describes the standard phases and major processes of the New York State System Development Lifecycle (SDLC), using a common language and in sufficient detail to enable a Project Manager to plan and manage a system development project.

System Development Lifecycle Overview

The material in this section is organized according to a generic system development lifecycle. While no two development efforts are exactly alike, all projects should progress through the same six phases:

1. **System Initiation** – in which the Business Case and Proposed Solution developed during Project Origination are re-examined to ensure that they are still appropriately defined and address an existing organizational need. This validation effort provides the Project Team with the basis for a detailed schedule defining the steps needed to obtain a thorough understanding of the business requirements and an initial view of staffing needs. In addition, a high level schedule is developed for subsequent system development lifecycle phases.

2. **System Requirements Analysis** – in which the needs of the business are captured in as much detail as possible. The Project Manager leads the Project Team in working with the Customers to define what it is that the new system must do. By obtaining a detailed and comprehensive understanding of the business requirements, the Project Team can develop the Functional Specification that will drive the system design.

3. **System Design** – which builds upon the work performed during System Requirements Analysis, and results in a translation of the functional requirements into a complete technical solution. This solution dictates the technical architecture, standards, specifications and strategies to be followed throughout the building, testing, and implementation of the system. The completion of System Design also marks the point in the project at which the Project Manager should be able to plan, in detail, all future project phases.

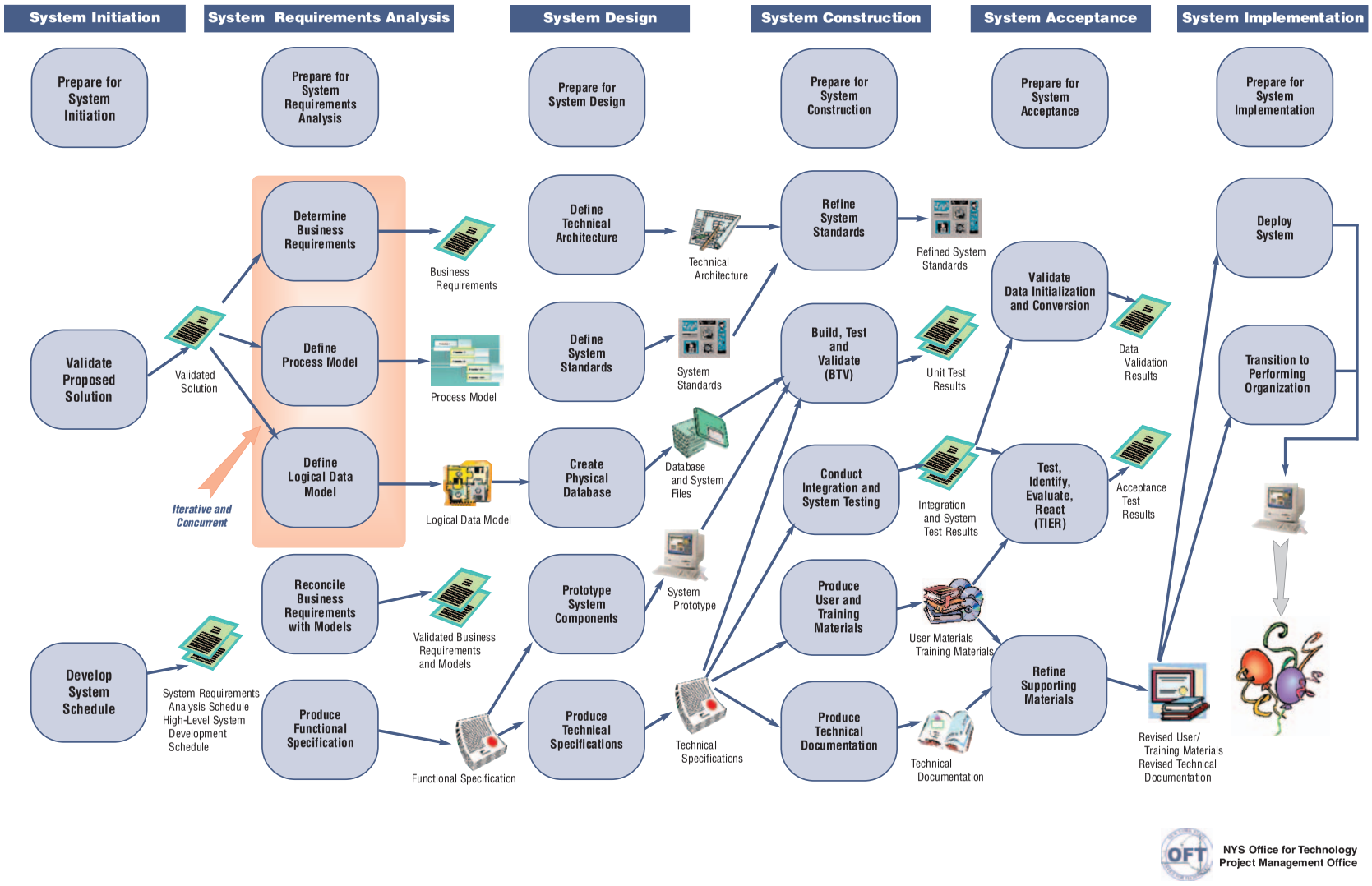
4. **System Construction** – throughout which the Project Team builds and tests the various modules of the application, including any utilities that will be needed during System Acceptance and System Implementation. As system components are built, they will be tested both individually and in logically related and integrated groupings until such time as a full system test has been performed to validate functionality. Documentation and training materials are also developed during this phase.

5. **System Acceptance** – during which the focus of system validation efforts shifts from those team members responsible for developing the application to those who will ultimately use the system in the execution of their daily responsibilities. In addition to confirming that the system meets functional expectations, activities are aimed at validating all aspects of data conversion and system deployment.

6. **System Implementation** – the final phase of the lifecycle, which comprises all activities associated with the deployment of the application. These efforts include training, installation of the system in a production setting, and transition of ownership of the application from the Project Team to the Performing Organization.

The following diagram illustrates every phase, process and deliverable in the system development lifecycle.

Figure 0-1 *NYS Project Management Guidebook* The System Development Lifecycle

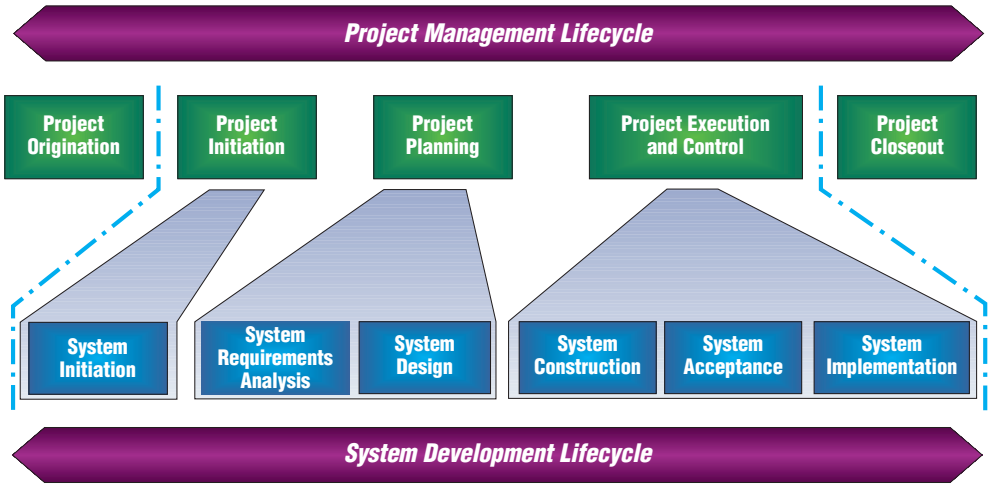


NYS Office for Technology
Project Management Office

Mapping the Project Management and System Development Lifecycles

The phases of the system development lifecycle generally align with the phases of the project management lifecycle; however, SDLC phases do not correspond one-to-one with the project management phases. One of the challenges for system development projects is aligning the SDLC with the project management lifecycle. The following diagram demonstrates how the phases of the two lifecycles may be integrated.

Figure 0-2



In reality, each phase of the SDLC can be thought of as a mini-project in itself, requiring planning, execution, and analysis. As the Project Team proceeds through the project, they will need to create a clear and detailed plan for the phase immediately in front of them, along with a higher-level view of all remaining phases. As the team executes each phase, they will collect additional information that will enable the detailed planning of subsequent phases. Some of this information will be a natural by-product of having performed the processes associated with the current phase (e.g., as the detailed technical design evolves throughout the System Design phase, the team will have a much better understanding of the modules that will need to be built during construction, and will therefore be able to refine any prior estimates and plans for System Construction). Additional information can be obtained through a focused

analysis effort, performed at the completion of each phase. This assessment is analogous in many respects to conducting the Post-Implementation Review as described in Section I, Project Closeout, although it is typically conducted in a less formal fashion. The responsibilities of the Project Manager include assessing how closely the phase met Customer needs, highlighting those aspects of the phase that worked well, identifying lessons learned and best practices in an attempt to derive ways to improve upon processes executed throughout the project, and, most importantly, communicating results.

The SDLC defined in this section may appear to have characteristics of a classic "waterfall" approach, which assumes that each phase and process is completed and agreed upon before the next phase begins. The reality is, however, that phases generally overlap, with each successive phase introducing changes to the work of the prior phase, resulting in an iterative process.

This SDLC is also consistent with newer techniques for system development, such as Rapid Application Development (RAD). RAD allows users to participate in an iterative design and development process. Conceptually, the project "loops" through the Design, Construction, and Acceptance phases, followed by re-Design, revised Construction, Acceptance, and so on. Project management deliverables such as the Project Scope Statement, Project Schedule, and budget estimates are refined to reflect increasing clarity of scope and requirements with each iteration.

While there is the potential to compress Requirements Analysis, Design, and Construction in RAD approaches, compression introduces increased risks. It is important, therefore, to include risk analysis in each iteration of the design, build, and evaluate loop. When a prototype is presented, Project Managers must actively and diligently address the management of Customer expectations and the maintenance of current documentation.

The RAD approach has advantages, since it usually achieves results quickly, the design is less abstract, and users have assurance that up-to-date requirements are considered. Its disadvantages include difficulty in controlling the process and ensuring the creation of an acceptable product.

Many factors may impact your choice of approach to follow when developing a system. The better you know your Customers and Stakeholders, and the better you understand the factors that may influence their assessment of the project, the more likely it will be that your approach will suit their personalities, preferences, vision, and needs.

The key is to pick the approach that you believe will provide the best complete solution, balancing the preferences of your Customers, the abilities of your Project Team, and the overall business drivers (legislated timeframes, overall time to market, etc.).

In any approach, the basic SDLC processes must be performed – what differs is the timing of their execution. As with the project management methodology, if processes or deliverables are skipped, the Project Manager must record the reasons why, and must describe how the objectives of that process/deliverable will otherwise be met.

Understanding the Breadth of System Development Projects

When assessing the scope of a system development project, it is important that the needs, goals, and challenges of the project are understood from many perspectives. The ***business requirements***, which define the high-level Customer objectives and vision for the system, are used to determine the scope of the system. When capturing the business requirements, it is essential that the Project Team look at all aspects of the system, including:

- **Functional Requirements** – describing processes and tasks that the Consumer must be able to accomplish through the use of the system. These can typically be categorized as processes that require action on the part of Consumers (data entry, selection of a system command, etc.), and those that are not directly related to human interaction with the system (for example, off-hours processing or the automated exchange of information between systems).
- **Technical Requirements** – identifying technical aspects and constraints that must be considered when defining the new system. Considerations may include accessibility needs of Consumers, whether or not the storage and

handling of data must follow specified encryption regulations, or whether the system will operate on internal agency hardware or will be hosted at either an internal or external data center.

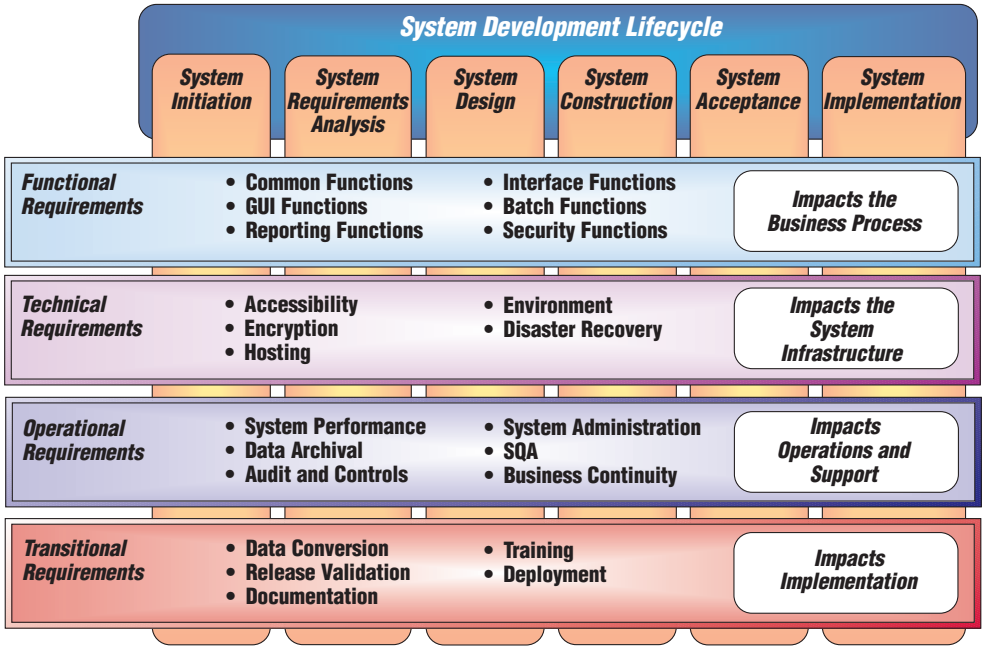
- **Operational Requirements** – specifying any administrative constraints or expectations that must be supported by the system. These requirements may include system performance expectations, technical infrastructure constraints, security mechanisms that must be followed, the need to regularly archive data, and any mandated audit and control processes.
- **Transitional Requirements** – defining the realm of conditions that must be satisfied prior to physically implementing the system in a production environment, or to relegating support responsibilities to the Performing Organization. Data conversion requirements and development and delivery of Consumer training programs and materials fall into this category.

Formally organizing thoughts along these four dimensions will drive the identification of tasks to be performed beginning in System Initiation and continuing throughout the lifecycle. The Project Manager is responsible for creating this broad view of requirements and communicating it to the Project Team, establishing a pattern that should be carried throughout all project phases.

The following diagram illustrates some representative categories of requirements that the team should consider when defining tasks and activities throughout the system development project.

Figure 0-3

Representative SDLC Considerations



It should be noted that not all considerations may be applicable to every project, and additional categories may be discovered that are not represented in the diagram. The fundamental point is that for those considerations that do apply to the project, there will be corresponding activities required throughout each and every phase of the SDLC, all contributing to the eventual implementation of the system. Regardless of whether the Project Team is performing System Initiation, Requirements Analysis, Design, Construction, Acceptance, or Implementation activities, they will need to understand and address the full realm of functional, technical, operational, and transitional requirements to ensure a successful project. In an attempt to reinforce this point, this diagram will be revisited in each of the individual SDLC phases that follow, drawing specific references to the processes relevant to each phase of the lifecycle.

Software Quality Assurance

In the way that project management provides the umbrella under which all project activities are directed, software quality assurance provides the foundation on which all system development activities should occur so that the highest quality system possible will be delivered. According to the IEEE Standard Glossary of Software Engineering Terminology, quality is defined as the degree to which a system, component, or process meets specified requirements and Customer needs and expectations.

As will be stressed throughout the following chapters, it should be noted that simply meeting requirements is not enough to guarantee a successful system development effort. Ultimately, Customer needs and expectations can be met only if the requirements are fully and correctly captured in the first place.

Analogous to the Quality Assurance Plan associated with the project management lifecycle, software quality assurance programs should be comprised of three components – quality standards, quality assurance processes, and quality controls.

Software Quality Standards define the programming standards, and development/testing standards to be followed throughout the project.

Software Quality Assurance Processes define practices and procedures to be used by the Project Team to meet the quality standards, and to provide management with evidence that these procedures are being followed.

Software Quality Controls comprise a series of reviews and audits that evaluate deliverables with respect to defined standards and acceptance criteria. These controls include software testing techniques and peer reviews.

The key to these SQA efforts is that they must be performed throughout all phases of the project. In addition, all SQA efforts should ideally be performed by a third party, independent from the team members responsible for delivering the system. Availability of staff and budget are two factors that must be considered in determining the feasibility of applying an independent SQA Analyst or team to the project. In developing the

overall system development plan, the Project Manager needs to allocate sufficient time and resources to perform the appropriate level of SQA activities, and must obtain management commitment to providing these resources as called for in the Project Schedule.

Project Roles and Responsibilities

As presented in the Section I Introduction, Project Roles and Responsibilities, there are many groups of people involved in both the project and project management lifecycles. When staffing system development projects, there are a number of roles that should be considered. It should be noted that the SDLC only provides details to the phase and process level, whereas the PM lifecycle further decomposes activities down to individual tasks. As a result, while the roles identified within the SDLC are representative of those that are typically required in a system development effort, the function of the role as it relates to a given SDLC process may not be specifically described within that process narrative.

The **Project Team** consists of a Project Manager and a variable number of Project Team members who are responsible for planning and executing the project. Team members specific to the System Development Lifecycle are described below.

The **Facilitator** leads sessions to identify business requirements and issues, keeps sessions focused and productive, draws out issues and ideas from all participants, and maintains clear and open communications within the session.

The **Business Analyst** effectively leads discussions with the Customers to determine the business requirements, participates in preparing the data and process models, prepares module specifications, test data, and user documentation materials, assists in prototyping activities, and develops strategies for testing and implementation.

The **Database Administrator** is responsible for providing and maintaining database administration policies and procedures, approving and executing database scripts, performing database tuning activities, and transforming a pictorial representation of the system data (the Logical Data Model) into physical database tables that support the final system.

The **Data/Process Modeler** develops and maintains data and process models to represent the business information needs in the area under study, develops and defines the data dictionary, validates models with the Customers, and participates in prototyping.

The **Technical Lead/Architect** drives the logical process and data models into an application architecture, establishes architecture guidelines, and develops strategies for the creation and distribution of applications.

Application Developers include all those responsible for developing prototypes, technical specifications, and application code, and for executing test scripts.

The **Software Quality Assurance (SQA) Analyst** is responsible for establishing and executing the Quality Assurance Plan, for assisting in the preparation of test scripts and test data, and for participating in integration and acceptance testing efforts.

Technical Services (HW/SW, LAN/WAN, TelCom) include all those responsible for the ordering, installation and maintainence of hardware and software components, LAN/WAN components and telecommunications components.

The **Information Security Officer (ISO)** is responsible for identifying and enforcing security standards and processes.

Technical Support (Help Desk, Project Administration, Documentation, Trainers) includes all those responsible for supporting the development of the new system. Support includes the documentation of user, training, operation materials, and help files, training for Customers, responding to technical and business questions forwarded to the Help Desk, and supporting the project and associated administrative processes.

Figure 0-4 New York State System Development Life Cycle Templates

Phase	Template	Description	Page in Text	Page in Appendix
System Requirements Analysis	Business Requirements Document	A document containing detailed functional, technical, operational and transitional requirements for the system being developed	45	99
System Requirements Analysis	Functional Specification	A document describing the logical grouping of related processes and the mapping of those processes to business requirements and data items.	59	103
System Design	Technical Architecture	A document describing the system architecture in terms of hardware, software, tools and peripherals, and the distribution of system components and processes across this architecture.	81	109
System Design	System Standards	A document detailing the standards to be applied and adhered to throughout the project.	87	115
System Design	Technical Specifications	A compilation of system diagrams, module specifications, and test plans that serve as a detailed, comprehensive blueprint for the system.	109	121
System Construction	Defect Log	A document used to log defects encountered when performing integration, system, data validation or acceptance testing, and track their resolution.	145	131

1 SYSTEM INITIATION

Purpose

The purpose of **System Initiation** is to validate the Proposed Solution developed during the Project Origination phase of the Project Management Lifecycle, and to estimate the system development effort in greater detail. In this phase, the broad parameters of the new system are defined, and applicable system development activities are identified.

Once the overall approach has been confirmed, it is necessary to estimate the effort and resources required for the next phase in elemental detail, and to provide high-level estimates for subsequent phases, to the extent necessary to support the project management lifecycle deliverables and activities of Project Initiation.

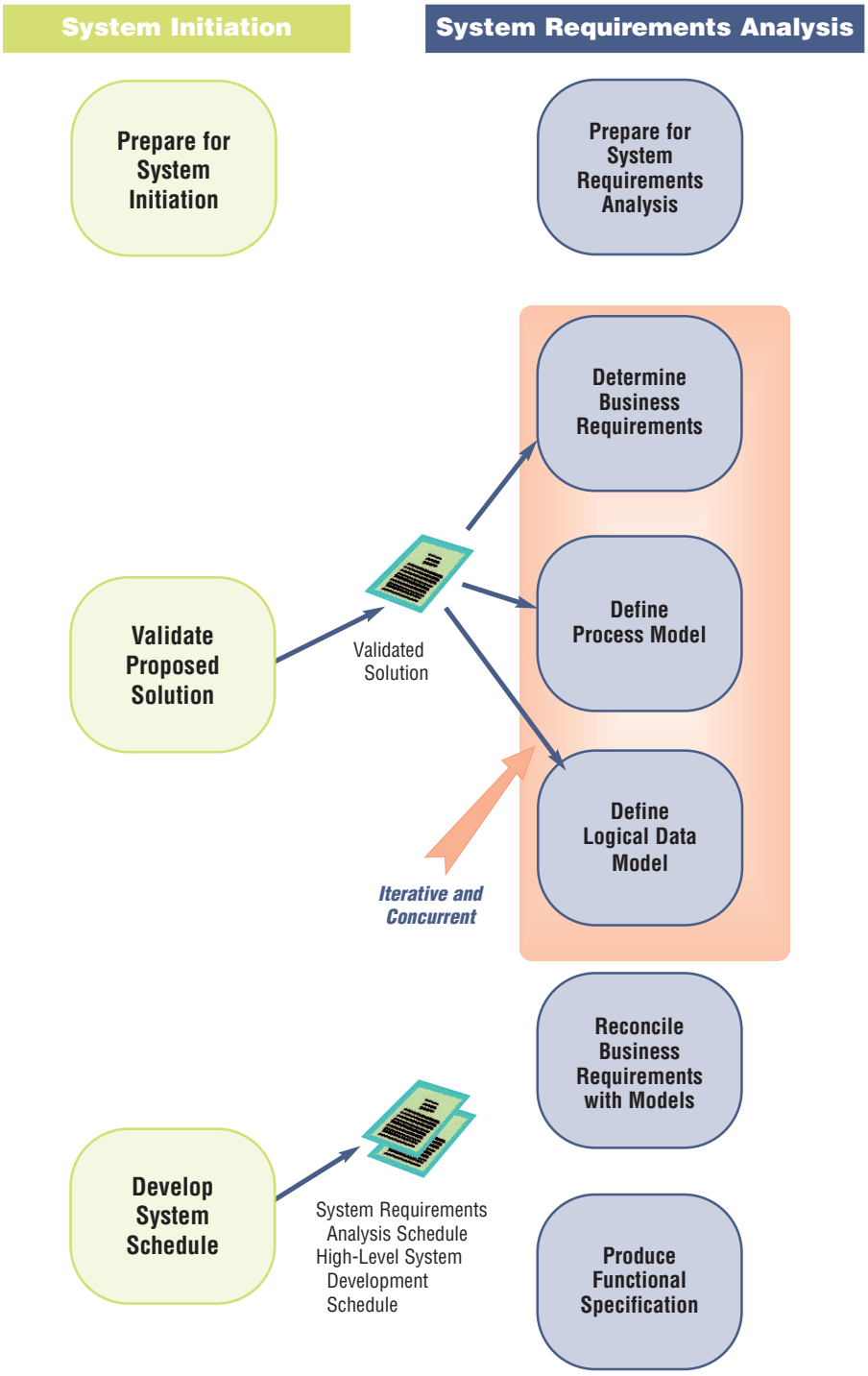
List of Processes

This phase consists of the following processes:

- **Prepare for System Initiation**, where the initial members of the Project Team familiarize themselves with the project's defining documents and plan the activities for the rest of the phase;
- **Validate Proposed Solution**, where the original technology direction and system development approach are validated;
- **Develop System Schedule**, where a detailed System Requirements Analysis schedule is developed, and a high-level system development schedule is produced.

The following chart illustrates all of the processes and deliverables of this phase in the context of the system development lifecycle.

Figure 1-1


System Initiation
System Requirements Analysis
Prepare for System Initiation
Prepare for System Requirements Analysis
Determine Business Requirements
Define Process Model
Define Logical Data Model
Validate Proposed Solution
Validated Solution
Iterative and Concurrent
Reconcile Business Requirements with Models
Develop System Schedule
System Requirements Analysis Schedule
High-Level System Development Schedule
Produce Functional Specification

List of Roles

The following roles are involved in carrying out the processes of this phase. Detailed descriptions of these roles can be found in the Introductions to Sections I and III.

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Lead

List of Deliverables

The following table lists all System Initiation processes, some techniques available for use in executing these processes, and process outcomes and deliverables.

Figure 1-2

Processes	Techniques	Process Deliverables *(Outcomes)*
Prepare for System Initiation	Interviews Document Gathering and Reviews	*Established Team and Environment for System Initiation*
Validate Proposed Solution	Brainstorming Research	Validated Solution
Develop System Schedule	Brainstorming Research Estimating	System Requirements Analysis Schedule High-Level System Development Schedule

1.1 PREPARE FOR SYSTEM INITIATION

Purpose

The purpose of **Prepare for System Initiation** is to ensure that the Project Team, and the environment in which it will operate, are ready for successful completion of this phase.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Lead

Description

In addition to the Project Manager, Business Analyst and Technical Lead roles are required to complete the team for this phase.

Identification, assignment and orientation of new team members required for this phase are activities in Project Initiation in the project management lifecycle.

Environment preparation includes gathering all relevant project and historical documentation, and placing it in the document repository. At a minimum, the Project Team should have available the Project Proposal (consisting of the Business Case and Proposed Solution), and any relevant evaluation and decision documentation. Any historical data, such as best practices or performance statistics from similar earlier efforts, can also serve to guide the Project Team towards – or away from – certain solutions.

A record of prior efforts to develop a similar system (including current system documentation, if it is a replacement) can also be very helpful, although you should take care not to pre-judge the approach either because of, or in spite of, prior experiences.

1.2 VALIDATE PROPOSED SOLUTION

Purpose

The purpose of **Validate Proposed Solution** is to make sure that the original technology decision and system development approach still represent the optimal solution for the identified business need.

Description

Considerable time may have elapsed since the Project Proposal (and its constituent Proposed Solution) was developed, due to the vagaries of the budget process or to other procedural delays. In the interim, the Performing Organization may have changed its course and the state-of-the-art technology may also have changed significantly. With rapid advances in technology, it is certain that the longer the period of time between the original proposal and the commencement of System Initiation, the more likely it is that the chosen technology is no longer supported, has become obsolete, or commands a vanishing talent pool.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Lead

If System Initiation closely follows the development of the Project Proposal, it may not be necessary for the Project Team to perform all parts of this validation process.

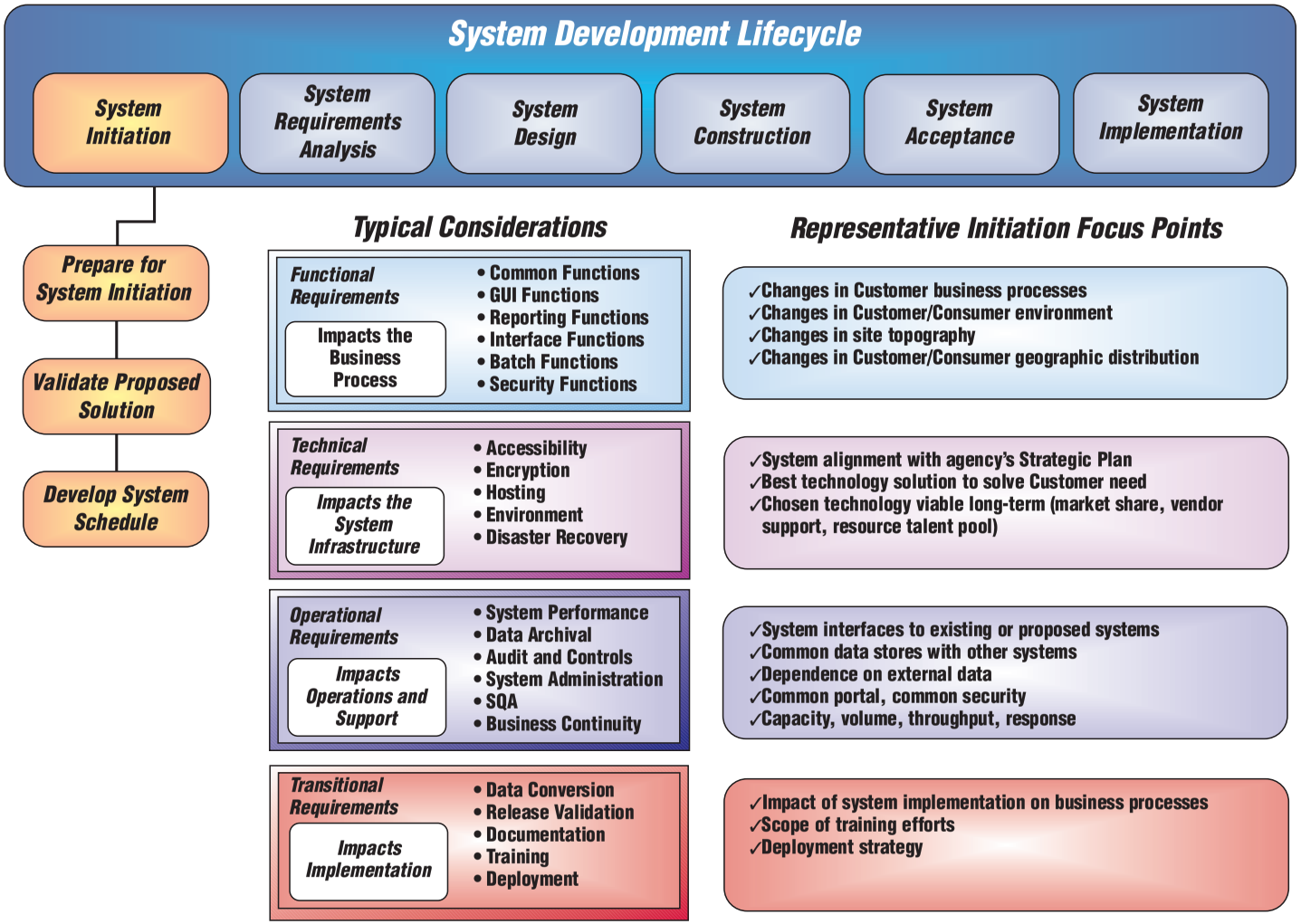
To validate the Proposed Solution, the Project Team must:

- understand the agency's current Strategic Plan, and how the new system fits into it;
- assess the proposed technology solution in view of Customer needs, the Performing Organization's long term technology direction and constraints, and state-of-the-art technology; and
- confirm feasibility of the proposed system development approach.

In assessing the Proposed Solution, the team must consider how it fits into the Performing Organization's application portfolio of existing or proposed systems. A System Context Diagram can be used to illustrate how the new system will make use of, or add to, existing major data stores, and how it will interface with other systems identified in the Strategic Plan (extract, update, data entry, etc.).

The next step is to confirm that the Proposed Solution is still the best option for the current set of business needs and conditions (as they are reflected in the Project Charter project management deliverable). For example, reorganization may have dispersed Consumers over a large geographical area, necessitating a re-evaluation of the originally proposed technology that was best suited to many Consumers in close physical proximity to one another.

Figure 1-3 System Initiation Considerations



In deciding whether the proposed technology direction represents an industry trend or a dead end, there are numerous professional journals available by subscription or free on the Web. There are forward-looking reports by organizations such as Gartner, Inc. or the Meta Group, Inc., and many consulting companies can offer valuable advice. The NYS Office for Technology can also serve as an authoritative point of reference.

Once it has been determined that the proposed technical solution fits into the Performing Organization's Strategic Plan, any lingering questions may be resolved through formal reviews, or directed to the Project Sponsor.

If it becomes apparent that the original Proposed Solution is no longer the optimal one, the team should propose an alternative solution. The Project Sponsor will then direct the team to proceed with the original solution, to take the alternative proposal, or to take action such as terminating the project or repeating the project management lifecycle starting at an earlier process or phase.

Finally, the system development approach needs to be validated against the latest understanding of both the business needs and the technology solution. Certain decisions must be considered even if they cannot yet be made. Among them are:

- Should the system be developed in-house or acquired as a Custom Off-The-Shelf (COTS) solution?
- Are there available resources to develop/customize the system, or is it necessary to contract for additional resources?
- Is the choice of technology platform predicated on the existing environment, or does the system offer an opportunity to upgrade the infrastructure?

If the Proposed Solution involves using infrastructure and/or services provided by the NYS Office for Technology, early notification to OFT is necessary to ensure smooth integration with planned service upgrades and other service demands. The Office for Technology may also be able to provide valuable contacts in other agencies where systems similar in function or technical components have been developed.

Deliverable

- **Validated Solution** – The team should update the original Project Proposal, or recreate the Proposed Solution using the template from Section I, Project Management Lifecycle.

1.3 DEVELOP SYSTEM SCHEDULE

Purpose

The purpose of **Develop System Schedule** is to create a detailed schedule for System Requirements Analysis and a high-level schedule for the remaining phases.

Description

After the technical solution has been validated, it is possible to decide how the rest of the System Development Lifecycle will be applied to this particular system development effort.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Lead

Using a project scheduling tool of choice and referring to the chart of the System Development Lifecycle, the Project Manager should populate the project with the phases and processes and the suggested roles. Then the Project Team, including the Business Analyst and the Technical Lead, should walk through and brainstorm each process, attempting to answer as well as possible the following questions:

- What are the system development deliverables in this process?
- What are the tasks necessary to produce this deliverable?
- Is there a logical way to organize the tasks associated with this process for this system? That is, are the modules already defined? Will they be worked on in parallel or serially?
- How complex is each task likely to be?
- What skills are required to perform each task?
- How many resources with each identified role/skill set will be needed to produce the deliverables?

In addition to thinking through the deliverables necessary for implementing the desired functionality, the team should consider the technical, operational and transitional requirements of the system (refer to Figure 1-3). These additional requirements will influence the definition of necessary tasks and the tasks' order or complexity.

This scheduling process needs to be performed to an elementary level of detail for the next phase, but only at a high level for the subsequent phases. The goal of the high-level estimating process is not premature precision, but rather a coherent, purposeful system development strategy that will form the basis for subsequent efforts (understanding that it will nevertheless be changed, augmented and enhanced throughout the lifecycle.)

At the end of System Initiation, all system-related materials gathered and produced by the Project Team should be consolidated and prepared for use as input for the next phase.

Deliverables

- **System Requirements Analysis Schedule** – Task-level schedule for the System Requirements Analysis phase of the System Development Lifecycle.
- **High-Level System Development Schedule** – Process-level schedule for the remaining System Development Lifecycle phases.

Both the detailed and high-level schedules produced in this process are part of, and are integrated into, the High-Level Project Schedule deliverable of the Project Initiation phase of the Project Management Lifecycle.

Measurements of Success

The success of this phase is measured by how readily the team can perform the next phase. The schedule for System Requirements Analysis should be immediately executable by the team.

The Project Manager can assess how successfully the project is proceeding by utilizing the measurement criteria outlined below. More than one "No" answer indicates a serious risk to the next phase and the eventual success of the system.

Figure 1-4

Process	Measurements of Success	Yes	No
Prepare for System Initiation	Have you obtained the materials that describe (1) the business needs and benefits, by functional unit; (2) the proposed solution; (3) the reasons that this project and this solution were selected for initiation?		
Validate Proposed Solution	Does the head of the information technology organization (or designate) agree that the system solution fits into the Strategic Plan?		
Develop System Schedule	Have the standard development procedures been customized for the specific system components defined for this system, including functional, technical, operational and transitional requirements?		
	Do you have management commitments for team member availability for the next phase?		
	In the High-Level System Development Schedule, do you know if the effort allocated to system development phases correlate to industry-accepted norms?		

Phase Risks / Ways to Avoid Pitfalls

PITFALL #1 – PLAYING WITH TOYS

You read the original Proposed Solution, and it seems very astute and on the ball: it promotes the latest technology trend, and is full of impressive-sounding concepts.

You consult with your technology folks, and they enthusiastically endorse the approach: it's the latest rage, and they just can't wait to get their hands on those great new packages. In fact, they will use your system as the showcase of the new technology, and allocate all kinds of resources to make it succeed. All of them will just be delighted to work on your system.

Their enthusiasm is contagious, and you consider yourself one lucky Project Manager. Until, that is, you realize that none of them actually knows how the new technology works; not that it stops them from trying to use it, all in their different ways, until they make a total mess of your "sandbox".

People like gadgets and gismos, buzzwords and catch phrases, and the technology folks like them most of all. They are like a bunch of little kids, going all googly-eyed at a new shiny toy.

So before you agree to play with a new solution, make sure that somebody convinces you that the tried-and-true really won't work here, and that you need to take on all the risks of the new technology. Do your own research, consult with outside experts, talk to the managers who've implemented technology projects successfully in the past, do the risk vs. reward analysis, and dull that "bleeding edge".

PITFALL #2– UNDERESTIMATING REQUIREMENTS ANALYSIS

If you have worked in your organization for a long time, you may have dealt with your Customers long enough to know what they want better than they know it themselves. In your mind, it all seems so clear – they need access to this data, here's how they will want to see it broken down, and this is the type of interface they are most comfortable with. In fact, you are probably already constructing the screens and reports in your mind, visualizing their reaction to this great new technological advance. The real challenge seems to be in the development, while the requirements analysis effort is often seen as a

straightforward exercise – get together with some key individuals, tell them what the new system will do for them, and you are done.

The reality, of course, could – and should – be very different. The estimated effort includes not only identifying the requirements of ALL interested parties, but also documenting and reconciling them, to a point where the Functional Specification could be passed to another team altogether and still result in an accepted product.

Frequently Asked Questions

How do I map the Project Team roles identified in the Section I, the Project Management Lifecycle, to the people working on the project? Do I need that many people on the project?

First of all, you need to understand that a role does not necessarily translate to an FTE. Depending on the size and complexity of the system you are trying to develop, the size and geographical distribution of your Customer and Consumer base, and finally on the versatility of your Project Team, you may need many people to fill one role, or you may have one person fill many roles. The following steps should help you map available people to required roles:

1. Determine Project Team requirements. Using the High-Level Project Schedule, estimate which roles are going to be required throughout the duration of the project, and to what extent.
2. Understand Stakeholder landscape. Using the Description of Stakeholder Involvement (contained in the Project Plan), refine your list of required roles, adjusting for approach (individual vs. group requirements meetings, individual vs. classroom training), geography (all Customers in one building vs. multiple facilities throughout the state) and size (a handful of Customers who are also Consumers vs. scores of representatives from competing Customer and Consumer groups.)
3. Research your team capabilities. Understand people's skills, interests and proclivities. Document their work hours and availability to the project.

4. Map roles to people using all information available. Document and address any apparent gaps.

I have a few skeptical Customers who, because they've been around for a while, have the SME (Subject Matter Expert) status on my project. However, they've never participated in a formal system development effort before. How do I get them to contribute in a productive fashion?

The good old carrot and stick approach should work: on the one hand, appeal to their self-interest and flatter them immoderately; on the other, make sure their manager is on board with the project and is aware of their expected contributions.

You should be able to convincingly demonstrate to any Customer (and most Consumers) why and how the new system will benefit them; in addition, you should genuinely appreciate (and praise!) their knowledge of the business process.

Having an experienced Facilitator and/or Business Analyst is extremely helpful as well, to engage all participants in a constructive dialog and set realistic expectations for future contributions.

What are the go/no go points in the SDLC? How do they integrate with decision points in the project management lifecycle?

The decisions to proceed with, or to halt, the system development effort properly belong in the project management lifecycle. To that end, the final process in each of its first three phases (Project Origination, Project Initiation and Project Planning) contains an explicit go/no go decision point. However, even before the project comes to those points, it is entirely possible that the system development lifecycle will necessitate project go/no go decisions.

The first such event may occur in System Initiation. If it is determined that the original Proposed Solution is no longer adequate or desirable, it may be prudent to halt the system development process (and the project) altogether, or re-initialize the project back at the Project Origination phase.

The next event may occur during System Requirements Analysis. If the business requirements gathered during this phase push the scope of the project way beyond the initial estimate, it may be necessary to halt Project Planning activities until the amended scope, schedule and budget are approved, or the decision is made to terminate the project.

The same is true for System Design. Insurmountable technical difficulties or irreconcilable differences over the prototype may jeopardize the success of the project, disrupting the flow of Project Planning and forcing a go/no go decision.

Finally, the controls put in place for Project Execution and Control may be triggered by difficulties with System Construction, Acceptance and even Implementation activities, and an abrupt end (or reiteration) of the Project Management Lifecycle may occur as a result.

How do I estimate the System Requirements Analysis phase?

The first thing to remember is not to do it in a vacuum. Use your entire team to flesh out the answer.

Start by decomposing the SDLC and creating the Work Breakdown Structure for the System Requirements Analysis phase. Consider the Stakeholder landscape and your team's capabilities and availability (see the first question above) and map out to whom you will be talking, when, how many times, and for how long. Then, based upon your (and your team's) knowledge of the business environment, estimate how much effort – and time – you will need to come up with all the System Requirements Analysis deliverables. Be as precise as possible, decomposing each process into elementary tasks, and each deliverable into its constituent (or pre-requisite) work products.

Is this SDLC appropriate for outsourced/contracted out engagements? How do I know what system development methodology my vendors will use?

The whole point of creating common project management and system development methodologies for New York State is to have a consistent approach to system development on ALL engagements. Not only does it streamline planning and execution and enable state Project Managers (and Project Team

members) to move within and among state agencies with a minimal learning curve, it also provides a standard for agency staff to use when contracting with private vendors. The state can now provide the methodology for its contractors, and direct them to adhere to it, instead of requiring New York State staff to adjust to the different development methodologies of each firm with whom they contract.

This SDLC is designed to be generic enough for virtually all system development efforts, and allows utilization of nearly all platforms, tools and techniques.

2 SYSTEM REQUIREMENTS ANALYSIS

Purpose

The purpose of **System Requirements Analysis** is to obtain a thorough and detailed understanding of the business need as defined in Project Origination and captured in the Business Case, and to break it down into discrete requirements, which are then clearly defined, reviewed and agreed upon with the Customer Decision-Makers. During System Requirements Analysis, the framework for the application is developed, providing the foundation for all future design and development efforts.

System Requirements Analysis can be a challenging phase, because all of the major Customers and their interests are brought into the process of determining requirements. The quality of the final product is highly dependent on the effectiveness of the requirements identification process. Since the requirements form the basis for all future work on the project, from design and development to testing and documentation, it is of the utmost importance that the Project Team create a complete and accurate representation of all requirements that the system must accommodate. Accurately identified requirements result from effective communication and collaboration among all members of the Project Team, and provide the best chance of creating a system that fully satisfies the needs of the Customers.

The primary goal of this phase is to create a detailed Functional Specification defining the full set of system capabilities to be implemented, along with accompanying data and process models illustrating the information to be managed and the processes to be supported by the new system. The Functional Specification will evolve throughout this phase of the SDLC as detailed business requirements are captured, and as supporting process and data models are created, ensuring that the eventual solution provides the Customers with the functionality they need to meet their stated business objectives.

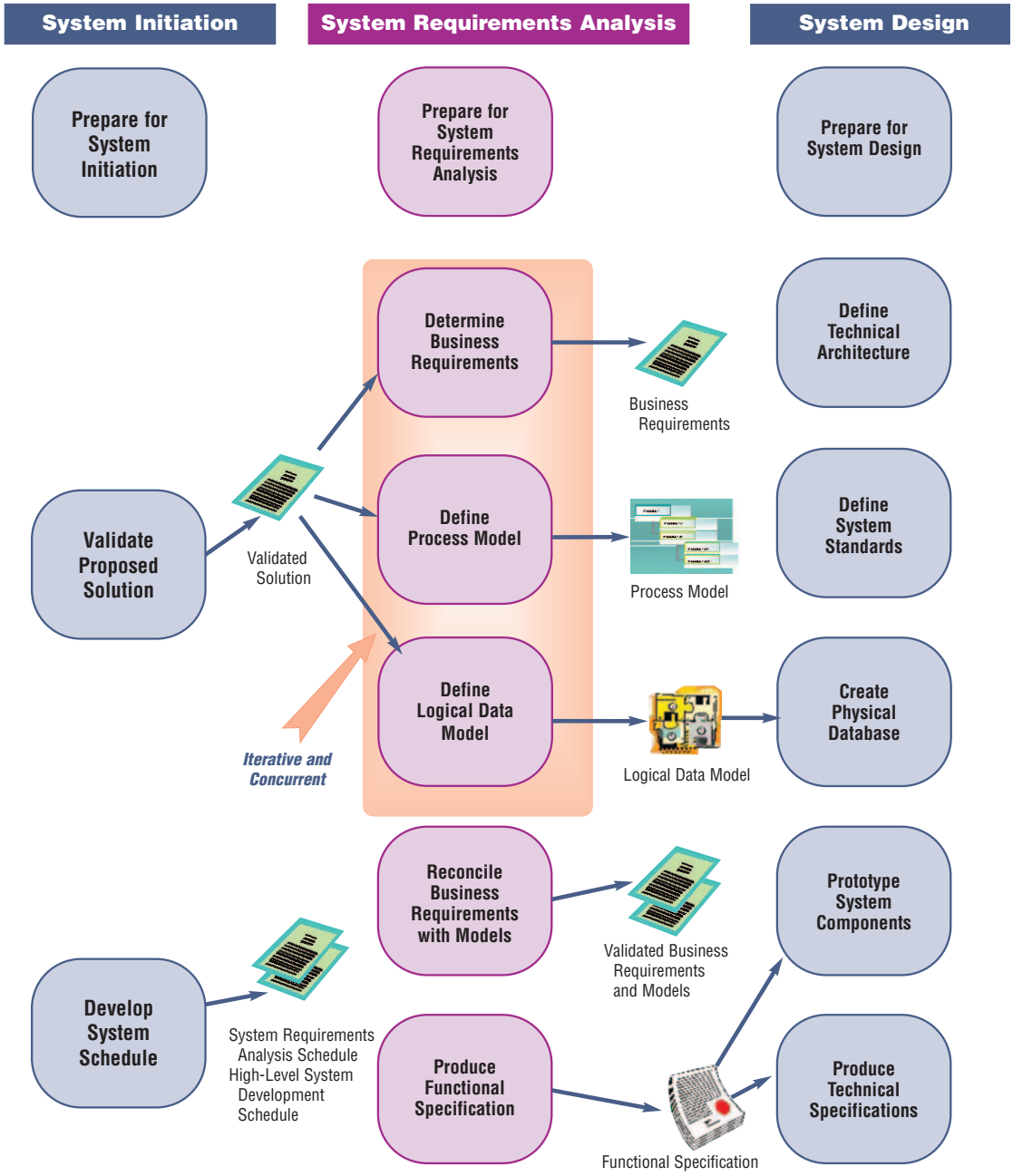
List of Processes

This phase consists of the following processes:

- **Prepare for System Requirements Analysis**, where steps are taken to ensure that the project environment and Project Team members are adequately prepared to both capture and analyze the system requirements;
- **Determine Business Requirements,** where in-scope and out-of-scope business requirements are identified, business rules are defined and documented, and interfaces to and from the new application are discussed;
- **Define Process Model,** where a pictorial top-down representation of the major business processes that interact with the system is diagrammed and decomposed into manageable functions and sub-functions until no further breakdown is feasible;
- **Define Logical Data Model,** where data that supports the processes and business rules is logically modeled, identifying entities and their relationships to other entities, and defining attributes with their business definitions;
- **Reconcile Business Requirements With Models,** where the Project Team ensures that the Process and Logical Data Models accommodate all requirements and business rules;
- **Produce Functional Specification,** where interfaces, processes and data are merged to describe systematically how the Consumer will use the application, and how data will be retrieved, processed and stored.

The following chart illustrates all of the processes and deliverables of this phase in the context of the system development lifecycle.

Figure 2-1


System Initiation
System Requirements Analysis
System Design
Prepare for System Initiation
Prepare for System Requirements Analysis
Prepare for System Design
Determine Business Requirements
Define Technical Architecture
Business Requirements
Validate Proposed Solution
Validated Solution
Define Process Model
Process Model
Define System Standards
Define Logical Data Model
Create Physical Database
Logical Data Model
Iterative and Concurrent
Reconcile Business Requirements with Models
Prototype System Components
Validated Business Requirements and Models
Develop System Schedule
System Requirements Analysis Schedule
High-Level System Development Schedule
Produce Functional Specification
Produce Technical Specifications
Functional Specification

List of Roles

The following roles are involved in carrying out the processes of this phase. Detailed descriptions of these roles can be found in the Introductions to Sections I and III.

- Project Manager
- Project Sponsor
- Facilitator
- Business Analyst
- Database Administrator
- Data/Process Modeler
- Technical Lead/Architect
- Software Quality Assurance (SQA) Analyst
- Technical Services (HW/SW, LAN/WAN, TelCom)
- Information Security Officer (ISO)
- Technical Support (Help Desk, Documentation, Trainers)
- Customer Decision-Maker
- Customer Representative
- Consumer
- Performing Organization
- Stakeholders

List of Deliverables

The following table lists all System Requirements Analysis processes, some techniques available for use in executing these processes, and process outcomes and deliverables.

Figure 2-2

Processes	Techniques	Process Deliverables *(Outcomes)*
Prepare for System Requirements Analysis	Team Skills Assessment Site Walk-throughs Technology Needs Assessment Tool Needs Assessment	*Established Team and Environment for Requirements Analysis*
Determine Business Requirements	Interviews JAD Sessions Brainstorming Storyboarding Critical Success Factor Interviewing Context Diagramming Use Case Diagramming Prototyping Walk-throughs Potential Problem Analysis Expressing Logic: Pseudo Code, Structured English, Object Oriented Logic	Business Requirements
Define Process Model	Work Flow Diagramming Flow Chart Diagramming Process Modeling Customer Event Diagramming Use Case Diagramming Decision Trees Prototyping	Process Model
Define Logical Data Model	Entity Relationship Diagramming Data Normalization/ De-Normalization	Logical Data Model
Reconcile Business Requirements With Models	CRUD Matrices Gap Analysis	*Analysis Assessment* Validated Business Requirements and Models
Produce Functional Specification	Process Association and Grouping Logical Organization Work Flow Clustering Expressing Logic: Pseudo Code, Structured English, Object Oriented Logic	Functional Specification

2.1 PREPARE FOR SYSTEM REQUIREMENTS ANALYSIS

Purpose

The purpose of **Prepare for System Requirements Analysis** is to position the Project Team and their working environment to ensure successful completion of System Requirements Analysis. This is the point at which the Project Team prepares to capture the detailed functional, technical, operational, and transitional requirements of the system.

Description

In preparing for this phase, the Project Manager must focus on the Project Team and the environment in which the team will work.

With each new project phase comes the need for new skills, experience, and, potentially, new Project Team members. The team needed during this phase must possess analytical skills that allow them to continually "peel the onion", driving to continuously deeper levels of requirements definition. Experience in effective interviewing, facilitation, various modeling techniques, requirements gathering, and gap analysis will be extremely beneficial.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- Customer Decision-Maker
- Customer Representative

In reviewing the Validated Solution all team members must share a clear and common understanding of the scope of this phase of the project, the Project Schedule, the deliverables to be produced, and their individual responsibilities relating to the creation of these deliverables.

Regardless of the size of the development effort being undertaken, System Requirements Analysis may place the greatest demand upon Customers in terms of resources and the extent of their required participation. During the preparation for this phase, the Project Manager should continue to manage the Customer's expectations surrounding this participation. Less involvement typically leads to a less acceptable finished product. In addition, many

individuals earmarked to participate in the requirements gathering sessions may not have been privy to earlier project scope-setting sessions. This can lead to the possible perception of these upcoming sessions as opportunities to identify or request functionality and features that are beyond the original intent of the project. Since management of scope creep is an essential role of the Project Manager, this may be an appropriate time to review the established change management processes with the Customer.

At the start of the System Requirements Analysis phase, it is the Project Manager's responsibility to ensure that the environment in which the Project Team will work is properly established. Beyond the obvious need to ensure that team members have adequate equipment to perform their duties, there are additional elements of the environment that should not be overlooked. The project repository, a secure area for maintaining work products and deliverables that was established during Project Initiation, continues to evolve over subsequent phases of the project. Although the establishment of the repository itself is important, it is equally necessary to define the mechanisms and processes to be followed for creating and maintaining all System Requirements Analysis related materials.

2.2 DETERMINE BUSINESS REQUIREMENTS

Purpose

In **Determine Business Requirements**, information is gathered from a variety of project participants relating to the vision and scope of the system. From this, specific detailed requirements are identified and documented so that they address the stated business need. These requirements are then decomposed into a set of business rules.

Description

While this process specifically addresses the capturing of Business Requirements for the new system, the reality is that it may be necessary, and is often beneficial, for the Project Team to determine these requirements while simultaneously defining the supporting process and data models. By conducting these three processes (Determine Business Requirements, Define Process Model, and Define Logical Data Model) concurrently, as opposed to sequentially, the team can develop the process and data models as information and requirements are defined, and can update these models as a result of gathering new or changed information.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

Because the three processes are performed not only concurrently, but also often iteratively, it is important for the Project Team to tightly manage the documentation for each process so that requirements are not lost, misunderstood or overlooked. The Project Manager may need to utilize techniques and/or tools to help document requirements and ensure that they are not missed. The Glossary contains a brief description of some of the techniques available to the Project Manager; examples include storyboarding, interviews, joint application design sessions (JAD), Unified Modeling Language (UML), prototyping, data flow diagramming, process modeling, and entity-relationship diagramming.

Don't hesitate to use one of the commercially available tools to assist the Project Team in their documentation efforts. Through use of these tools, changes made to a process are automatically carried through to all other associated processes or business rules.

Determining Business Requirements requires eliciting, analyzing, specifying, prioritizing, verifying and negotiating business functions that the system must deliver and support. The results are captured in a Business Requirements deliverable (use Figure 2-5, Business Requirements template, as a guide). During this process it is important to have all of the Stakeholders involved. Since this is the process in which all business and processing requirements are determined and agreed to, it is critical that all parties understand the ramifications of including or excluding requirements from scope. This is an opportunity to work out business process issues as a group, in order to reach optimal performance and efficiency within an organization or even across organizations or functional areas. Decisions made will impact remaining phases, so all parties involved in the project lifecycle should be heard, and all areas of concern or question should be thoroughly addressed. Reaching consensus and agreement on the final deliverable from this phase will help to ensure that everyone gets the product to which they agreed.

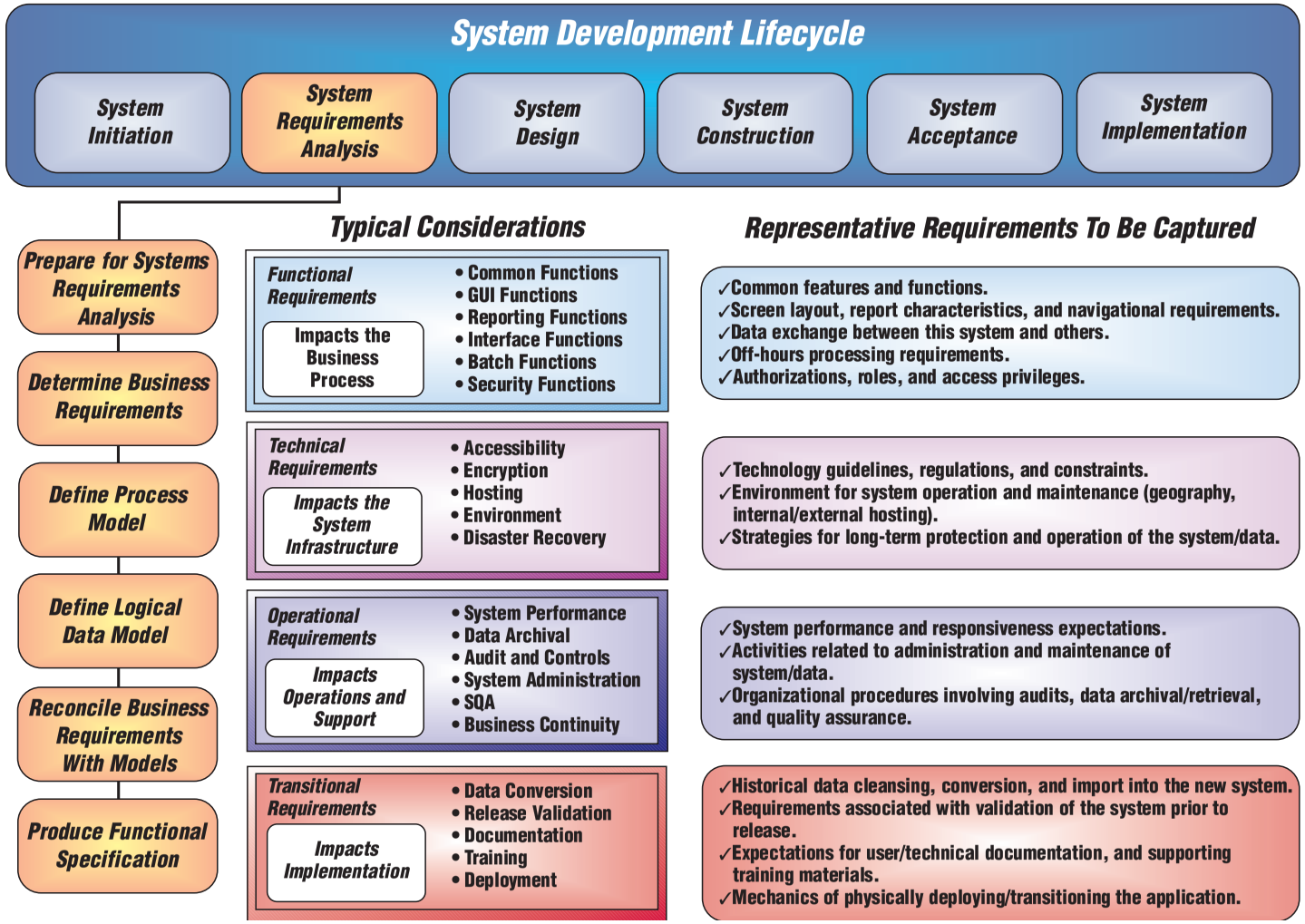
As stated in the SDLC Overview, requirements fall into multiple categories that, while related, are themselves separate and distinct. These categories are:

Figure 2-3

Category	Description
Functional Requirements	Requirements that define those features of the system that will specifically satisfy a Consumer need, or with which the Consumer will directly interact.
Technical Requirements	Requirements that identify the technical constraints or define conditions under which the system must perform.
Operational Requirements	Requirements that define those "behind the scenes" functions that are needed to keep the system operational over time.
Transitional Requirements	Requirements that define those aspects of the system that must be addressed in order for the system to be successfully implemented in the production environment, and to relegate support responsibilities to the Performing Organization.

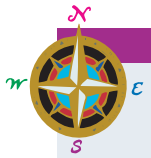
When capturing Business Requirements, the Project Manager must ensure that the Project Team addresses all of the categories above. Figure 2-4 illustrates the types of considerations and requirements that the Project Team must capture specific to System Requirements Analysis.

Figure 2-4 System Requirements Analysis Considerations



One approach to eliciting requirements from the Customer is to hold one or more JAD sessions. For these sessions, assembling individuals from both the program areas and IT into cross-functional groups can help clarify how proposed changes to a business process may impact operations. The benefit of having a session with multiple representatives from the program areas is that the pros and cons of business process changes are heard and discussed by all involved.

Requirements gathering, when properly facilitated, establishes a forum for everyone to be heard, for issues to be worked through, and for resolutions to be defined that meet the needs of all parties. Through this forum, multiple opinions may enhance the team's understanding of how certain processes are currently being performed, better defining how they should be structured within the context of the new application. This approach may also result in negotiations of functionality. There may need to be some trade-offs, and as a result processes may be reexamined and redefined. As the sessions progress, the Project Team must constantly assess and analyze the requirements.



A common mistake when coordinating the logistics of interviews or group requirements definition sessions is to hold these meetings where the majority of the participants represent only the Customer and Consumer populations. Doing so may set the stage for the Project Team to form a type of tunnel vision in which business requirements that focus primarily, or even exclusively, on the functional aspects of the system are captured. Just by the nature of their day-to-day responsibilities, Customers will often approach these requirements definition sessions from the perspective of, "What do I need this system to do in order for me to perform my duties?" Many operational, technical, and transitional requirements do not fall within the answers to that question. It is up to the Project Team member running these sessions (typically a Business Analyst or Facilitator) to make certain that these aspects of the system are also discussed, and that the individuals best positioned to provide this information are represented at the requirements definition meetings.

It is not unrealistic to assume that there may come a point at which negotiation or consensus building activities will break down, and that resolution of an issue may require insight or information not available to the Project Team. Ultimately, there must be a single decision making body responsible for resolving such issues. A key role of the Project Sponsor or Customer Decision-Maker is to make the final determination regarding these issues, and to communicate the decision to the Project Team. The Project Sponsor may or may not choose to share the

rationale for such decisions with the entire team, nor is it guaranteed that the team will agree with determinations that are made. The Project Manager should encourage the team to support the decisions and move forward.

The following steps should assist the team in building a useful and comprehensive Business Requirements document:

Absorb the requirements. Before the requirements can be analyzed, they must first be collected. Team members need to be sponges and take everything in, no matter how unimportant or inconsequential it may seem. There are many approaches to gathering these requirements, but all start with effective listening. Regardless of the technique used, the Project Team must remember that the goal of this process is to understand what the Customers need – not what the team members think they need.

Interpret the requirements. Now that the requirements have been captured, the Project Team must think about them. What have they heard? What was missed? Look for unanswered questions or contradictions. The requirements must be stated as clearly and concisely as possible, avoiding combining requirements and eliminating subjective wording. If the system must do A and B and C and D, there should be four distinct and verifiable requirements that can each be approved or rejected on its own merits. Avoid ambiguities and opinions. If the requirement is that some process should be "easy", the team should go back and find out exactly what that means. What would make something about that process difficult or more complex than it should be?

Bind the requirements. While defining what the business requirements are, it is also necessary to determine what they are not! This is done to establish consensus on the Project Scope and to clarify any scope issues. At least some requirements that were captured will be labeled "out of scope."

Categorize the requirements. Even for a relatively small system, you are likely to end up with scores of requirements. To understand how they relate to each other, and to effectively deal with them later on in the process, it is necessary to separate them into categories, logically grouping the requirements according to related business functions or organizational boundaries.

Prioritize the requirements. Regardless of how accurately the business requirements reflect the business need, it may not be feasible to implement them all at once (or even at all). In finalizing the requirements to be implemented, it will be necessary to prioritize them according to their criticality to the business. This classification into core, essential, and desirable will very likely involve both the Project Sponsor and Customer Decision-Makers.

The following guidelines may be used: "Core" requirements are the ones without which the system may as well not be developed at all; it will be of no use to most Customers without these. "Essential" requirements are those for which a short-term work-around could be developed (or for which an old process can hobble along for a little while longer) but over the long run, they have to be there. "Desirable" requirements are the "bells and whistles" that may be precious to certain constituencies, but without which the system will function just fine.

To put it another way, the system must go into production with all Core and a good portion of Essential requirements represented, and with a plan to implement the remaining Essential requirements in the subsequent phase.

Deliverable

- **Business Requirements** – A document containing detailed requirements for the system being developed. These requirements define the functional, technical, operational, and transitional capabilities, restrictions, and features that must be provided by the new system.

Figure 2-5 Business Requirements Document

< Name of Agency >

Business Requirements Document

< System Name >

Agency	
Project Name	
Project Sponsor	
Project Manager	
Document Date	
Prepared By	

*Enter the name of the **Agency** for which the system is being developed.*
*Enter the **Project Name**, and the names of the **Project Manager** and the **Project Sponsor**.*
*Enter the **Date** as of which this document is current.*
*Enter the names of the Project Team members by whom the document was **Prepared**.*

Figure 2-5 (Continued)

Business Requirements Document

TABLE OF CONTENTS

*The **Table of Contents** should be at least two levels deep.*

1.0 DOCUMENT SCOPE

***Document Scope** describes the goal of this document in a narrative.*
Example:
To define a consistent set of business requirements for the <XYZ> system and to identify what is in and out of scope.

The narrative should also provide an overview of the efforts conducted to gather business requirements.
Example:
This document summarizes the requirements gathered in a series of <X> Joint Application Design (JAD) sessions that were conducted by <members of the Project Team> with <DEF> and <GHI> business units between <date> and <date>.

2.0 GENERAL REQUIREMENTS

*The **General Requirements** section lists high-level business requirements that apply to the whole system.*
Example:
This system will provide a central repository for all <XYZ> data.

It should also include those requirements that are common to all Customer groups.
Example:
This system will provide ad hoc reporting capabilities to each Consumer business unit.

***NOTE:** By default, all requirements listed in this section are deemed to be Core to the system. Those general requirements that do not meet these criteria should be listed below under "4.0, Business Requirements Not Being Implemented".*

Figure 2-5 (Continued)

Business Requirements Document

3.0 SPECIFIC REQUIREMENTS

*The **Specific Requirements** section lists business requirements specific to each Customer group. A concise and specific listing of **Business Requirements** by **Business Function** should be provided. Requirements should be categorized, bulleted, detailed, and prioritized. These requirements should encompass the multi-dimensional aspects of the system (i.e., the functional, technical, operational, and transitional requirements).*

3.1 Business Unit

Description

*The **Description** identifies the purpose and main functions of the **Business Unit**.*

3.1.1 Business Function 1

Description

*In addition to **Business Function Description**, the narrative focuses on the desired business processes that will be in place when the new system is implemented as opposed to the current state of the business function, which can be documented, if necessary, in the Appendix.*

- Business Requirement 1 (Priority)
- Business Requirement 2 (Priority)
- Etc.

3.1.2 Business Function 2

Description

- Business Requirement 1 (Priority)
- Business Requirement 2 (Priority)
- Etc.

This format is repeated for all Customer and Consumer groups and their respective functions that require interaction with the system.

Example:

<XYZ> Unit

Description:
This unit is responsible for developing the <ABC> deliverable, maintaining the <DEF> function and executing the <GHI> process.

Business Function:
Developing the <ABC> Deliverable
Description:
The <XYZ> unit personnel will utilize the system reports to compile and produce the <ABC> deliverable (see attached).

Business Requirements *(Priority)*

1. *The System will provide a report detailing <JKL> expenditures that will be used for Page 2 of the <ABC> Deliverable.* *(Essential)*
2. *Etc.*

Figure 2-5 (Continued)

Business Requirements Document

4.0 BUSINESS REQUIREMENTS NOT BEING IMPLEMENTED

This section specifies requirements that will NOT be part of the new system.

Example:
Director of Contracts and Director of Claims have determined that the following functions are outside the scope of this system:
1. *<ABC> Process*
2. *<DEF> Deliverable*
3. *Etc.*

APPENDIX A – Requirements Definition Supporting Details

All published work products (meeting notes, session results, etc.) of individual interviews and group facilitated sessions held to determine business requirements should be included in this section.

A table detailing dates, times, topics and participants of all interviews and sessions should precede the compilation.

2.3 DEFINE PROCESS MODEL

Purpose

The purpose of the **Define Process Model** process is to create a pictorial representation of the functions and operations (i.e., the processes) that will eventually be performed by the system being developed.

Description

The second of the three concurrent processes within System Requirements Analysis, Define the Process Model may begin at any time after the Project Team has started collecting specific business requirements. The resulting Process Model of the system, also often referred to as the "To Be" model, illustrates the system processes as they are envisioned for the new system. Over time, this pictorial top-down representation of the major business processes will be decomposed into manageable functions and sub-functions until no further breakdown is possible. When combined with the detailed set of Business Requirements and the supporting Logical Data Model, this Process Model should completely address not only the full list of business needs to be satisfied by the new system, but also the vision for how the new system will provide and support this functionality.

Roles

- Project Manager
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

During the Determine Business Requirements process, a picture of the *current* business processes and practices will begin to evolve. This can be a useful tool in confirming that all current processes have been identified, and can be used by the Project Team as a means of ensuring that their Process Model has not neglected any existing functionality. There is a risk, however, that too much focus on current business processes may cause Customers to take a myopic view of the their true business needs, ultimately defining a system that provides little value over the system that is already in place.

A key to successful process modeling is to find a way to get your Customers to look beyond "what they do and how they do it", and to instead describe "what they need and how it could best be accomplished" if they were not forced to perform their duties within the constraints or limitations of existing systems and processes. One of the most common mistakes made during this phase of the development lifecycle is to automate a bad process, simply because that's the way the business has always operated.

Remembering that much of System Requirements Analysis is iterative, the Project Manager must ensure that as requirements are updated as a result of continued efforts to Determine Business Requirements, the Project Team also refines the Process Model to accommodate those changes.

The Project Manager must ensure that the Stakeholders and Customers periodically validate the Process Model as it is being developed. It is important that they understand that the Process Model is a representation of the proposed business solution, an attempt to meet everyone's needs. As part of validating the final Process Model deliverable with the Customer, it may be beneficial to conduct walk-throughs to map the defined business requirements to the diagrammed Process Model. A walk-through helps to identify any requirements missed by both the Project Team and the Customer, and helps to further validate that the requirements and processes are accurately decomposed.

Before the Customer accepts the final deliverable for this process, ensure that he or she understands the ramifications of acceptance. For instance, if a process critical to the application was overlooked in a JAD session, and therefore not modeled, and the deliverable has been approved, change control may be necessary. Ensuring that the processes have been identified and decomposed will make it more likely that the data model built to support the process is adequate. If a process has been overlooked, there will most likely be an impact to the design of the data model, and therefore to the database itself. This could definitely warrant change control downstream. This applies to the Business Requirements and Logical Data Model deliverables as well as the Process Model. You may need to validate these three deliverables incrementally during this phase, and provide the Customer with a final walk-through at the conclusion of this process, before proceeding to the development of the Functional Specification.

Deliverable

- **Process Model** – A graphical representation of the decomposition of all business processes that interact with the system.

2.4 DEFINE LOGICAL DATA MODEL

Purpose

The purpose of **Define Logical Data Model** is to identify all uniquely distinguishable objects either used or produced by the system (the data entities), to capture all of the characteristics that help define those objects (the data attributes), and to describe the relationships between the entities.

Description

Like process modeling, definition of the data model can start as soon as the interviews or JAD sessions begin. A Data Modeler is most often responsible for designing the logical representation of the data to support the business need. Typically, this model will evolve throughout the iterations of capturing and documenting the business requirements.

Roles

- Project Manager
- Business Analyst
- Facilitator
- Database Administrator
- Data/Process Modeler
- Technical Lead/Architect
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

The Data Modeler may begin to work on one of two paths: the first assumes that the application is brand new, and that the Data Modeler is working from a blank slate. In this case, informational requirements are captured as they are identified during the JAD sessions or interviews. As sessions are held, a view of potential entities and attributes is constructed and organized.

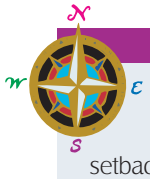
The second path assumes that the new application is going to replace an existing system. In this case, the Data Modeler may work with the current Data Base Administrator (DBA) to reverse engineer the existing database or file structures, often using a modeling tool. This enables the Project Team to use the existing structures as a starting point for the new Logical Data Model, and as a means

of validating that all informational needs of the system are being accommodated for in the Process Model.

It is important to define the entities and attributes in business English to facilitate Customer consensus and to ensure consistency with the organizational nomenclature forming the framework for the design of the application and the database. As additional requirements are flushed out during the interviews or sessions, the informational needs of the system are continually re-analyzed and re-applied to the data model. It is important to keep in mind, when identifying data sources, that consideration must be given to enterprise data and standard data that may be maintained in external systems (e.g., County Code Table, OSC code tables, etc.)

A key to successful data modeling is to ensure that the logical data model is not dependent on how the system processes the data. This ensures that data is grouped and organized based strictly on the informational needs of the system, and not based on an implied or assumed usage of the data by the system. The benefit is that the integrity of the data model will remain intact even if future business needs change the functionality of the system.

Defining the data model also helps to define the business rules by establishing the data entities (tables) and identifying attributes (fields). With the requirements and business rules known, and the Process Model outlined, the Project Team can begin to establish relationships between the data entities. This becomes the foundation of the data repository (or physical data model). As attributes are identified, the Data Modeler begins to build the Data Dictionary – again, in business English. Data normalization, a process in which complex relationships are simplified, is important once the Data Dictionary has been established. This eliminates redundancy, creates stable data structures, prevents anomalies, and simplifies data maintenance. The Logical Data Model is the basis for the DBA to create the physical database, so it is important that the Data Dictionary is clear in its definitions, and that all the data has been modeled appropriately.



The experience that a seasoned Data Modeler can bring to the Project Team can often make the difference between a successful project and one that encounters multiple setbacks or surprises. Understanding how to identify entities and attributes, establish relationships between the entities, normalize the attributes and define the Data Dictionary are important to developing a high performing application. These activities lay the groundwork for the technical team to build the physical database.

The Project Manager's goal for this process is to ensure that the Project Team accurately reflects the data requirements as they are defined, and as they relate to the Business Requirements and Process Model. The Project Manager should ensure that a process exists for the various Project Team members to share information and refine the requirements and models without risk of losing information, or jeopardizing the consistency and inter-dependencies of these deliverables.

The evolution of the Logical Data Model is analogous to the creation and confirmation of the Process Model in that it requires frequent interaction and validation by the Customer. While many of the deliverables in System Requirements Analysis present information in formats and terminology with which the Customer is familiar, this may not always be the case with data models. As a result, the Project Manager should anticipate the need for additional interaction during these reviews to ensure that the Customer can accurately interpret the output of this process. In addition, walk-throughs must be conducted in conjunction with, or in close proximity to, the reviews of the Process Model to be most effective.

Deliverable

- **Logical Data Model** – Diagrams and Data Dictionary information defining data elements, their attributes, and logical relationships as they align within a business area, with no consideration yet given to how data will be physically stored in actual databases.

2.5 RECONCILE BUSINESS REQUIREMENTS WITH MODELS

Purpose

The purpose of **Reconcile Business Requirements With Models** is to ensure that all business requirements and rules that have been captured have been accurately reflected and accommodated for in the Process Model and the Logical Data Model.

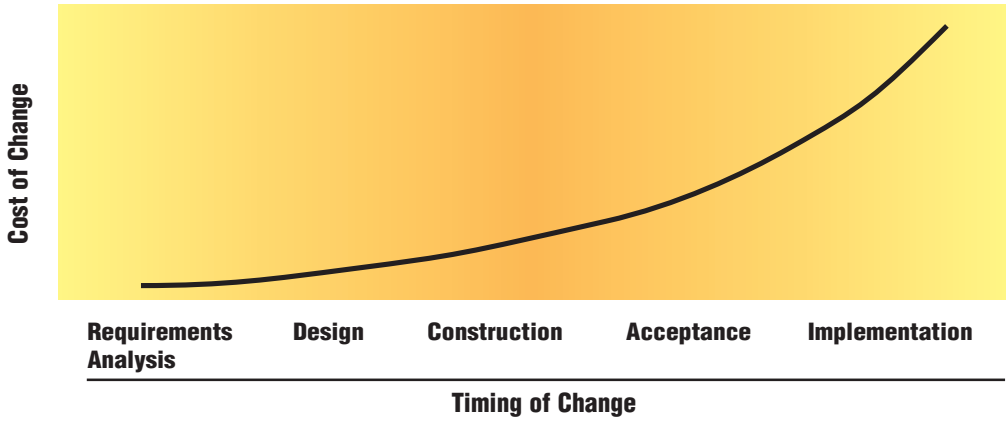
Description

Since the Process and Logical Data Models will ultimately form the basis of the system design, it is critical to invest the time here in System Requirements Analysis to ensure that these models are complete and accurate. If business requirements have been identified that are not reflected in these models, or if discrepancies exist between these models, then it is almost certain that this step will be revisited at some point later in the project. As Figure 2-6 indicates, the further out in the project that deficiencies in the business requirements are identified, the more costly the effort required to correct these deficiencies.

Roles

- Project Manager
- Business Analyst
- Facilitator
- Database Administrator
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

Figure 2-6 Impact of Change on Project Costs



A typical technique at this point in the SDLC is to perform an analysis assessment, which validates and cross-references all requirements to the process and data models, and which continues until all gaps have been identified, resolved, or recognized as an out of scope item.

One technique to reconcile the Business Requirements, Process Model and Logical Data Model is for the Business Analyst to create a gap analysis checklist or matrix that may be used to display the interactions among the requirements, data entities and the processes. This will help to ensure that all the requirements have been captured and modeled appropriately.

It is helpful to walk Customers through this exercise so that they understand how all requirements have been captured and modeled. These reviews are often iterative, and any gaps identified are corrected through subsequent revisions to the Business Requirements, the Process Model, or the Logical Data Model. It may be necessary to hold several review sessions to go over the reconciliation with different sets of Customers, remembering that the more people who review the output, the less likely it will be that key elements have been missed.

The Project Manager must ensure that the Customer understands the ramifications of overlooking a process, or of failing to decompose and model data appropriately. By understanding the potential impacts on both schedule and cost, the Customer is more likely to dedicate the appropriate staff to participate in these reviews.

Deliverable

- **Validated Business Requirements and Models** – An updated set of Business Requirements, Process and Logical Data Models that have been modified to address any gaps or weaknesses identified as a result of a gap assessment performed on these documents as a single unit.

2.6 PRODUCE FUNCTIONAL SPECIFICATION

Purpose

Produce Functional Specification maps the Logical Data Model and Process Model to the organizations and locations of the business. This process also produces the final deliverable for the phase – the Functional Specification.

Description

The ultimate goal of this process is to derive a comprehensive representation of the application that logically organizes related business processes, functions, data, and workflows. This provides a detailed roadmap from which the Customer Representatives can fully envision the final solution, and from which the Project Team can progress into the Design and Construction phases of the project lifecycle. Whereas all System Requirements Analysis efforts up to this point have been focused on continually decomposing information into discrete requirements or processes that can each be reviewed and validated on their own merits, this final process now builds a broader view of the system that groups the individual pieces of the solution into logically related business functions. The final result, the Functional Specification, defines and illustrates how each requirement of the system will eventually be satisfied in terms of business processes (or transactions).

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

Deliverables resulting from this phase of the lifecycle must capture the full set of requirements for the new system in a way that is completely independent of any development approach, methodology, or organizational constraints. Much like System

Initiation defined **why** the new system was needed, the System Requirements Analysis deliverables must define **what** the system must do without making any assumptions regarding **how** the system will be built.

The Functional Specification will present many views of the system, from different perspectives and at different levels of detail. For example, the System Context Diagram shows how the new system fits into the larger picture of the performing organization's application portfolio. The Business Flow Diagram shows how Customer and Consumer business units will interact with the new system from the business process and data flow perspectives. And the System Interface Diagram will present a view of the system from a perspective of Consumer interface, depicting menu structures and navigation paths of online system components, and organization and distribution of reports and other batch interfaces.

Within the Functional Specification, each business process or transaction will correlate to the set of Business Requirements that it satisfies and a representation of the corresponding data elements. The reports associated with each process, business constraints (such as related security or controls), interfaces to other systems and business functions, and any related administrative operations required to support the system should also be identified.

When identifying the new vision of the system with its proposed sets of related processes and functions, organizational or operational changes may be introduced. Ultimately, the Customer must be comfortable with these changes, and be willing and able to institute them. The Project Team must take this into account when establishing this framework. Both the Project Manager and the Stakeholders must find the appropriate balance between the potential need for the change, and the likelihood that it will be embraced throughout the organization.

As discussed in Determine Business Requirements, requirements gathering sessions frequently result in features or functions above and beyond those initially envisioned during Project Origination being identified. One advantage of a well-defined

Functional Specification is that it provides the Project Team with a vehicle to assist the Customer with decisions on trade-offs in functionality and scope, should the situation arise that sufficient budget is not available to support the development of the full set of capabilities.

To ensure that the Customer agrees with the final deliverable, the Project Manager should schedule a walk-through to review the concepts and flow of the Functional Specification, and to achieve consensus that the proposed grouping of processes defines a solution that will satisfy the Customer's needs.

Deliverable

- **Functional Specification** – Document describing the logical grouping of related processes and functions within the new system, along with the mapping of these processes to both the business requirements that they satisfy and the data items with which they interact.

Figure 2-7 Functional Specification Template

< Name of Agency >

Functional Specification

< System Name >

Agency	
Project Name	
Project Sponsor	
Project Manager	
Document Date	
Prepared By	

*Enter the name of the **Agency** for which the system is being developed.*
*Enter the **Project Name**, and the names of the **Project Manager** and the **Project Sponsor**.*
*Enter the **Date** as of which this document is current.*
*Enter the names of the Project Team members by whom the document was **Prepared**.*

Figure 2-7 (Continued)

Functional Specification

TABLE OF CONTENTS

*The **Table of Contents** should be at least two levels deep.*

1.0 DOCUMENT SCOPE

***Document Scope** describes the goal of this document in a narrative.*
Example:

To define a comprehensive set of functional specifications for the <XYZ> system and to identify what is in and out of scope.

The narrative also provides an overview of the efforts conducted to gather business requirements, derive process and logical data models, and to reconcile business requirements with these models.

2.0 GENERAL FUNCTIONAL SPECIFICATIONS

*The **General Functional Specifications** section details those specifications that are common to all aspects of the system (e.g., the menu structure, security, accessibility, overall performance requirements, etc.).*

Three graphical representations of the overall system should be included:

- *a System Context Diagram, showing how this system will interact with other existing systems,*
- *a Business Flow Diagram, showing how Customer and Consumer business units will interact with the system; and*
- *a System Interface Diagram, showing the application structure (menu structure and navigation of online components), organization of reports and other batch interfaces, and utilities.*

3.0 DETAILED FUNCTIONAL SPECIFICATIONS

*The **Detailed Functional Specifications** section lists functional specifications for each aspect of the system. The structure of this section is dependent on system organization. For example, if the system is organized to follow the business unit structure, with each sub-system supporting a specific Customer or Consumer group, then each **Sub-system Description** should list main characteristics and functions of that group; on the other hand, if the system is organized by type of interface (data entry, reporting, etc.), then the Sub-system Description should outline common characteristics of those system components.*

It may also be useful to provide more detailed versions of the Business Flow and System Interface diagrams for each sub-system.

Figure 2-7 (Continued)

Functional Specification

3.1 Sub-system

*The **Sub-system Description** describes the sub-system in a narrative.*

3.1.1 Component Type

*Depending on system structure (and Functional Specification document), it may be useful to organize system components by **type** (such as screens vs. reports, or tracking vs. auditing). If that is the case, **Component Type Description** would provide a rationale for such structural breakdown, and describe common elements of all components within that type.*

Component Type Description

3.1.1.1 Component 1

- Component Description
- Component Mockup (where appropriate)
- Component Business Flow
  - Cross-reference to Business Requirement(s), Logical Data and Process Models
  - Flowchart
  - Detailed Business Rules for each Flowchart element

*The **Component Description** should identify the appropriate Customer or Consumer group, and provide a description of how their needs are being met by this component.*
Where appropriate, a mockup of the component should be included in the document.
***Component Business Flow** details how the system supports the business process.*
*A **Cross-Reference** is provided to all prior deliverables.*
*A **Flowchart** details the system component's interaction with the business process. Every shape and arrow on the Flowchart is annotated with detailed descriptions of **Business Rules** governing that particular interaction or transformation.*

3.1.1.2 Component 2

- Component Description
- Component Mockup (where appropriate)
- Component Business Flow
  - Cross-reference to Business Requirement(s), Logical Data and Process Models
  - Flowchart
  - Detailed Business Rules for each Flowchart element

Figure 2-7 (Continued)

Functional Specification

4.0 OTHER SPECIFICATIONS

Besides functional aspects of the Business Requirements, the specifications for the system should also enumerate technical, operational and transitional aspects of the system.

4.1 Technical Specifications

This section documents in detail the technical specifications, regulations and existing constraints that must be considered in relation to business requirements. These include considerations such as accessibility, encryption, security, disaster recovery, and other technical areas.

4.2 Operational Specifications

This section should document in detail the operational specifications that must be considered in relation to business requirements. These include considerations such as system performance, data archival, audit and controls, system administration, software quality assurance and business continuity. The narrative should specify how these operational requirements may affect the organization and its current business processes.

4.3 Transitional Specifications

This section documents in detail the transitional specifications that must be considered in relation to business requirements. These include considerations such as data conversion, system testing, documentation, training and deployment. The narrative should describe how historical data will be cleansed, converted and imported into the new system, how expectations must be set for the deployment of and support of user and technical documentation and training, and what approach may be employed to physically deploy and transition the system into the organization.

5.0 BUSINESS REQUIREMENTS NOT BEING IMPLEMENTED

This section specifies those requirements that will NOT be part of the new system. If this list is identical to the one published in the Business Requirements document, a simple reference to the prior document may be substituted.

APPENDICES – SUPPORTING DOCUMENTS

The Appendices should contain all relevant documents provided by Customers and Consumers during System Requirements Analysis, as well as documents supporting decisions made while compiling the Functional Specification.

Measurements of Success

The immediate measurement of success for System Requirements Analysis is the acceptance of all deliverables by the Customer, while the ultimate measurement is whether or not the Project Team has created solid groundwork for the upcoming design and development of the application. Each process in this phase builds towards the final deliverable: Functional Specification. It is necessary to validate that certain steps have been successfully executed to ensure that the Functional Specification has been derived appropriately.

The Project Manager can assess how successfully this phase is proceeding by utilizing the measurement criteria outlined below. More than one "No" answer indicates you may have serious risk to the Project.

Figure 2-8

Process	Measurements of Success	Yes	No
Prepare for System Requirements Analysis	Have all new team members participated in project orientation sessions?		
	Is the team comfortable with the process defined for managing the deliverable repository?		
	Do all team members have experience with (or training on) the tools that will be used in this phase?		
Determine Business Requirements	Do the business requirements state what is in scope as well as what is out of scope?		
	Have the business requirements been reviewed and approved by the Customer?		
	Are the requirements stated in such a way as to allow for easy validation of their existence in the final solution (i.e., a Yes/No determination of whether or not the requirement has been satisfied by the new system)?		
	Are requirements prioritized within the Business Requirements deliverable?		
	Do the requirements consider the technical, operational and transitional aspects of the system, including elements such as: · Security/access needs; · Existing technical standards (accessibility, encryption, etc.); · Application hosting;		

Figure 2-8 (Continued)

Process	Measurements of Success	Yes	No
Determine Business Requirements (Continued)	· Disaster recovery; · Archiving, audit and regulatory needs; · Performance requirements by all Customers and Consumers for all aspects of the system; · Business continuity; · Data conversion?		
Define Process Model	Have all known business requirements and associated business rules been mapped and accommodated for in the Process Model?		
	Has the Process Model deliverable been reviewed and approved by the Customer?		
Define Logical Data Model	Have all known business requirements and associated business rules been mapped and accommodated for in the Logical Data Model?		
	Has the data model been normalized?		
	Has the Logical Data Model deliverable been reviewed and approved by the Customer?		
Reconcile Business Requirements With Models	Does the Customer agree that all aspects of the requirements and rules have been accommodated for in the process and data models?		
Produce Functional Specification	Has the Functional Specification deliverable been reviewed and approved by the Customer?		
	Has a system validation and testing approach been formulated?		
	Have training needs been identified?		
	Has an approach for the implementation and transition of the application been developed?		

Phase Risks / Ways to Avoid Pitfalls

PITFALL #1 – IF YOU BUILD IT (TOO SOON), WILL THEY COME?

Your Project Team is top-notch and you can't believe your luck in securing their stellar capabilities. You have the guru of gurus as your development lead. After the first JAD session he tells you he already has envisioned the application – it's a financial module and similar to one he has previously developed. He convinces you that he needs to skip the rest of the JAD sessions so he can begin to prototype immediately. He is so enthusiastic and convincing in his argument to you, that you bless him to go off and start creating. You have already re-forecast your

Project Schedule mentally and foresee an early delivery date. How sweet it is!

Four weeks into the JAD sessions, you haven't yet seen any output from him. He tells you he's almost there, just a few more tweaks … you remind him it is just a prototype, not a full-fledged construction effort yet. He assures you it is just what the Customer wants. But you realize that the Customer hasn't been involved in the prototyping exercise. How can your lead developer be creating anything without the Customer vision?

You arrange a meeting with the developer and the Customer to review the prototype as it is so far, and to elicit feedback. Your developer is giddy with anticipation, convinced that what he has already built will make their dreams come true. Then, the moment of truth: the developer begins to walk the Customer through the first screen, and it is already apparent that there is a huge disconnect between the developer's and the Customer's visions of the application. Productive dialogue quickly turns into heated debate.

It all could have been avoided if the developer's vision really was the Customer's vision. If only he had gone to the interviews and JAD sessions and heard the requirements, instead of making assumptions as to what HE thought the Customer should have. If you build it, you'd better make sure they can see it, or they won't come…..

PITFALL #2 – LET'S CROSS THAT BRIDGE WHEN WE COME TO IT

During your JAD sessions the Project Team has collected and prioritized a set of issues that need to be resolved by Customer Decision-Makers or a Steering Committee. However, some of the issues appear to be so huge that no one thinks they'll be able to resolve them. Instead of using the defined escalation process outlined in your Communication Plan, group consensus based on speculation and gut feeling is that by the time the Project Team is ready to start development, the issue will have been resolved by the 'higher ups'. Thus, the old saying, "Let's cross that bridge when we come to it."

Unfortunately, everyone's crystal ball that day was cloudy and/or cracked, and assumptions were proven wrong. The issue did not go away and it has suddenly become a high priority need because legislation was passed to 'make it so'.

Because the team did not address or resolve the issue, a major piece of functionality for the application is missing, and has a negative downstream effect on other modules. Why oh why didn't you escalate the issue when it arose, so you wouldn't be in the boat you're in now?

Taking the time NOW, during those JAD sessions, to address all issues, assess their impact and develop resolutions to them, is critical to achieving success in the ultimate design and development of the application. You will also save yourself hours of regret and heartache later.

PITFALL #3 – DESIGNING ON THE FLY

By nature, most technicians – including system developers – are natural problem-solvers. They are the kind of folks that disdain reading the twenty pages of directions, but immediately start fitting the parts together by eye; the kind of people who never read the manual, but just start pushing the buttons to see what happens.

When you sit with them at a requirements gathering session, you may see their eyes glaze over and their faces assume this far-away look. This is by no means due to a lack of interest, but because they are already designing – and may be even programming – the new system in their heads, based on some initial snatches of conversation.

Tell them to "Snap out of it!"

If the system requirements are not sufficiently defined and understood, the Project Team may experience "expectation gaps" – for example, the developers may build a Taj Mahal, while the Customer wanted a simple privy. At best, this may result in frustration and friction between the Project Team and the Customers – at worst, it can mean significant cost and schedule overruns that can impact many aspects of the Customer's business operations.

The trick to successfully capturing business requirements is to make sure that the Project Team does not get ahead of itself. The Business Requirements deliverable is a detailed and concise list that, without passing judgment and without in any way

indicating a solution, identifies the full set of business requirements that must be met by the system. If the list has been done correctly, the Project Team should be able to walk through it at any future point in the lifecycle and determine whether the emerging solution satisfies each requirement.

PITFALL #4 – GOT PICTURES?

Contrary to what the SDLC preaches, the wording used in Business Requirements can end up high-level, leaving many of the actual requirements open to interpretation. As this chapter has discussed, the decomposition of these requirements involves translating the business need into discrete, well-defined components that collectively provide the desired functionality. Therefore, you need to be creative in the ways in which these requirements are captured. Whenever possible, do not limit your understanding and representation of the requirements strictly to the written word.

As we all know, the English language is open to misinterpretation. How many of us have walked in for a haircut, described exactly what we're looking for, and walked out 30 minutes later wondering how "Take a little off the sides" turned into "I'd like you to make me as physically repulsive as possible, so that young children cry at the sight of me, and junkyard dogs turn and flee for safety"? In this case, a picture would have been worth more than a thousand words (at least to the children and junkyard dogs of the world).

The same is true when capturing system requirements. Words like "automate", "process", and "calculate" may mean different things to different people, and it is essential that everyone involved in the project share the same view of these requirements. Moving forward with an incomplete or incorrect vision of the system can be more terrifying than the haircut, except that this time, the ones running away will be your Customers. Never underestimate the power of prototyping, diagrams, illustrations, and modeling when trying to fully represent and validate the requirements that your Customer is communicating to you. These techniques, and the pictures that result from them, can be worth 1000 words, and many times that in dollars.

Frequently Asked Questions

What do you do if you don't have a Project Team schooled in the art of early life cycle techniques?

The skills required for the requirements definition and analysis activities are very different from those utilized in other phases of the System Development Lifecycle. The best strategy is to get an experienced Facilitator, if only for a few initial sessions, to transfer the skills to the team. However, if no early life cycle expert is available, you need to identify candidates among your team who are best suited to an interactive mode of communications, get them into some training, and have them study all available materials (including this chapter of the Guidebook). Then, make sure the team follows the SDLC methodology, and you, the Project Manager, follow the PM methodology to the letter, and rely on Project Sponsor and Customer Decision-Maker feedback to make sure your team is doing the right things.

How does your Project Team know when done is done?

In the normal course of events, if you have identified ALL Customers, Consumers and Stakeholders, gathered ALL their requirements as they relate to this system, built your Process and Logical Data Models, and then validated, cross-checked and verified everything, obtaining Customer and Project Sponsor approvals all along the way – then you are done! However, you are probably talking about cases when Customers keep changing their minds and you are trapped in an endless cycle of revisions and clarifications. Or cases when you keep chasing Customers who just won't spare a moment to talk to you. In either instance, think Time Box. With the help of your Project Sponsor, declare a deadline, communicate it to all participants, and end the game there.

What if you don't know which tools to select to help you through this phase?

While many tools exist that can simplify the creation of documents, illustrations, and other materials that support requirements analysis efforts, it should be stated that a pretty chart does not a good system make. People used to draw flowcharts by hand – and their systems did not come out any worse. The important thing is to get all the requirements, understand how

they relate to each other and to current and proposed business processes, and to get agreement and consensus on what will be delivered and when. And if in the process you can use some nifty tool and generate some neat documents – great!

What is the risk of just compiling all the process deliverables into one big deliverable, and foregoing the final deliverable: Functional Specification?

The Functional Specification deliverable has many aspects to it that its constituent parts miss (look at the annotated templates earlier in this chapter to see what they are). It is not enough to slap a bunch of work products together, tie them with a rubber band, and declare victory. "The whole is greater than the sum of its parts." The Functional Specification is a document that supplies the background for the effort and organizes the work products in a logical sequence that makes it easy to understand the process and verify the result.

How do I deal with a Customer who is afraid to commit?

We've all dealt with folks who believe that putting their signature on that acceptance form will forever doom them to dealing with an inadequate and cumbersome system; and whether that attitude comes from having such forms flung back into their faces in the past in response to reasonable requests or from genuine personality disorders is beside the point: you need to get the Customer OK, and he won't have any part of it! What to do?

For starters, don't say, "Just trust me!" (or equivalent). Like love, trust takes a while to form, and you can't really force it. Try to figure out what the underlying concern is. Is it fear of being locked into a particular design? Mistrust of the process used to gather the requirements? Lack of confidence in the players? Misunderstanding of the nature or purpose of the deliverable? Lack of knowledge about what will come next?

Education is the key here. Try to have a reasonable, calm conversation with the Customer Decision-Makers. Go over the process used to gather the requirements and prepare the deliverable. Explain the methodology, and the intent behind it: to support the Customer's business process with the best darn system a bunch of chip-heads can come up with.

And if they are still holding up the process without a good reason, pull out your big gun – the Project Sponsor – and have him earn his keep.

Where in the lifecycle do I define that a phone number is ten digits with dashes and parentheses?

Yes, it's a lot easier to record vague functionality requests ("the system should produce the required reports accurately and on time") than to get to brass tacks and figure out exactly what is going to happen, when and how.

When you are gathering business requirements, you should document every data source the Customer mentions, and mock up every interface the Customer requests. The data elements thus captured are formalized in the Data Dictionary (part of the Logical Data Model deliverable), and then further refined during requirements reconciliation, development of the Functional Specification deliverable, and creation of the prototype. By the time technical specifications are created, your data definitions should be set in concrete.

3 SYSTEM DESIGN

Purpose

The purpose of **System Design** is to create a technical solution that satisfies the functional requirements for the system. At this point in the project lifecycle there should be a Functional Specification, written primarily in business terminology, containing a complete description of the operational needs of the various organizational entities that will use the new system. The challenge is to translate all of this information into Technical Specifications that accurately describe the design of the system, and that can be used as input to System Construction.

The Functional Specification produced during System Requirements Analysis is transformed into a physical architecture. System components are distributed across the physical architecture, usable interfaces are designed and prototyped, and Technical Specifications are created for the Application Developers, enabling them to build and test the system.

Many organizations look at System Design primarily as the preparation of the system component specifications; however, constructing the various system components is only one of a set of major steps in successfully building a system. The preparation of the environment needed to build the system, the testing of the system, and the migration and preparation of the data that will ultimately be used by the system are equally important. In addition to designing the technical solution, System Design is the time to initiate focused planning efforts for both the testing and data preparation activities.

List of Processes

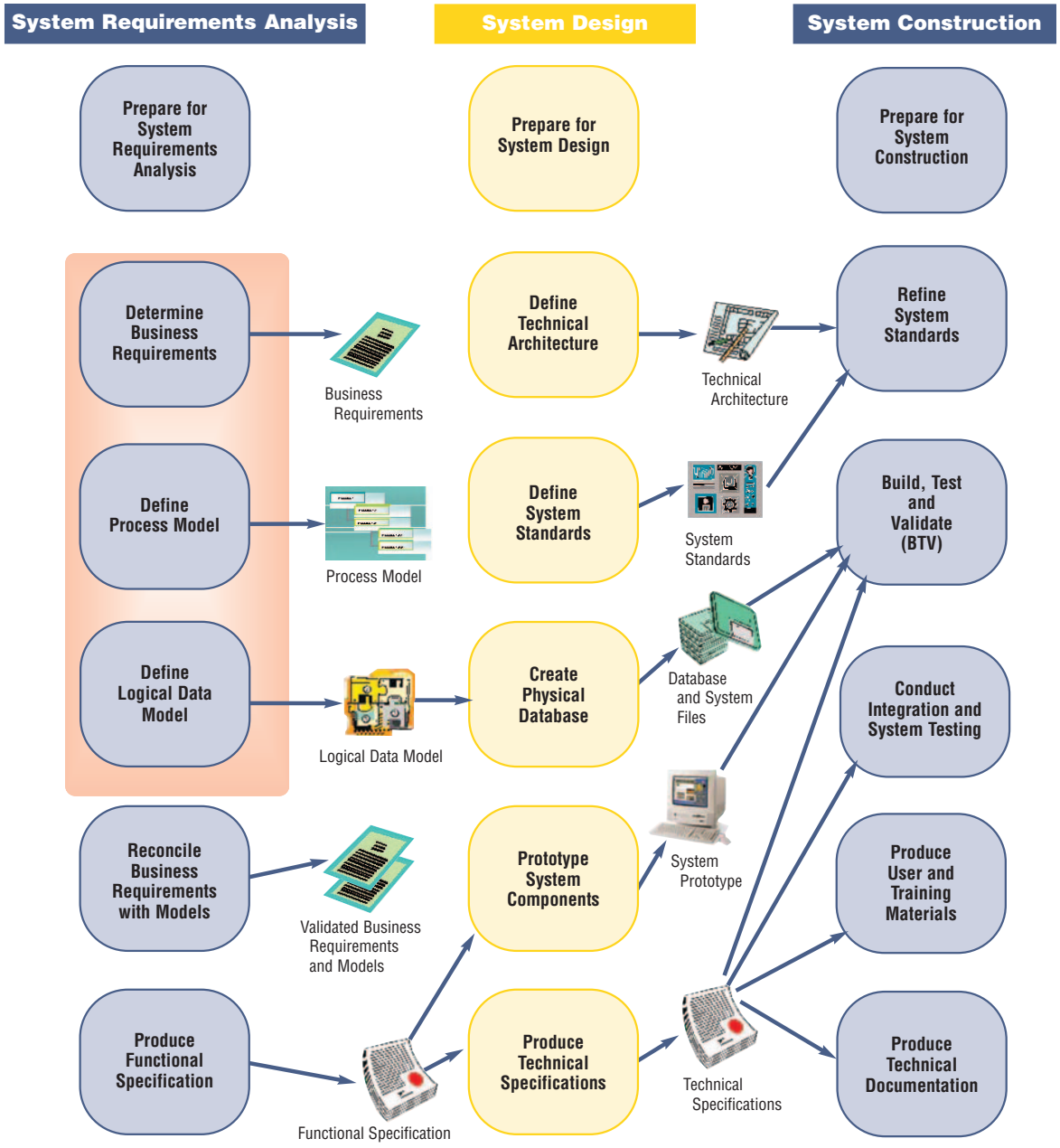
This phase consists of the following processes:

- **Prepare for System Design,** where the existing project repositories are expanded to accommodate the design work products, the technical environment and tools needed to support System Design are established, and training needs of the team members involved in System Design are addressed.

- **Define Technical Architecture,** where the foundation and structure of the system are identified in terms of system hardware, system software, and supporting tools, and the strategy is developed for distribution of the various system components across the architecture.
- **Define System Standards,** where common processes, techniques, tools, and conventions that will be used throughout the project are identified in an attempt to maximize efficiencies and introduce uniformity throughout the system.
- **Create Physical Database,** where the actual database to be used by the system is defined, validated, and optimized to ensure the completeness, accuracy, and reliability of the data.
- **Prototype System Components,** where various components of the solution may be developed or demonstrated in an attempt to validate preliminary functionality, to better illustrate and confirm the proposed solution, or to demonstrate "proof-of-concept."
- **Produce Technical Specifications,** where the operational requirements of the system are translated into a series of technical design specifications for all components of the system, setting the stage for System Construction.

The following chart illustrates all of the processes and deliverables of this phase in the context of the system development lifecycle.

Figure 3-1


System Requirements Analysis
System Design
System Construction
Prepare for System Requirements Analysis
Prepare for System Design
Prepare for System Construction
Determine Business Requirements
Define Process Model
Define Logical Data Model
Business Requirements
Process Model
Logical Data Model
Define Technical Architecture
Define System Standards
Create Physical Database
Technical Architecture
System Standards
Database and System Files
Refine System Standards
Build, Test and Validate (BTV)
Conduct Integration and System Testing
Produce User and Training Materials
Produce Technical Documentation
Reconcile Business Requirements with Models
Validated Business Requirements and Models
Prototype System Components
System Prototype
Produce Functional Specification
Functional Specification
Produce Technical Specifications
Technical Specifications

List of Roles

The following roles are involved in carrying out the processes of this phase. Detailed descriptions of these roles can be found in the Introductions to Sections I and III.

- Project Manager
- Project Sponsor
- Facilitator
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Application Developers
- Software Quality Assurance (SQA) Analyst
- Technical Services (HW/SW, LAN/WAN, TelCom)
- Information Security Officer (ISO)
- Technical Support (Help Desk, Documentation, Trainers)
- Customer Decision-Maker
- Customer Representative
- Performing Organization Management
- Stakeholders

List of Deliverables

The following table lists all System Design processes, some techniques available for use in executing these processes, and process outcomes and deliverables.

Figure 3-2

Processes	Techniques	Process Deliverables *(Outcomes)*
Prepare for System Design	Interviews Site Walk-throughs	*Established Team and Environment for System Design*
Define Technical Architecture	Interviews Document Gathering and Reviews Role/Authorization Analysis	Technical Architecture
Define System Standards	Interviews Brainstorming Policy and Standards Reviews	System Standards
Create Physical Database	Formal Walk-throughs Standard Data Definition Languages Data Administration Techniques (Data Normalization, De-Normalization)	Databases and System Files
Prototype System Components	Iterative Prototypes/Reviews Presentations GUI/Report Development Tools	Prototype and Proof of Concept Results
Produce Technical Specifications	Function Decomposition Expressing Logic: Pseudo Code, Structured English, Object Oriented Logic Operational Requirements Assessment System Load Analysis Business Impact Analysis Potential Problem Analysis Training Needs Decomposition	Technical Specifications

3.1 PREPARE FOR SYSTEM DESIGN

Purpose

Prepare for System Design formally marks the beginning of System Design and facilitates the transition from System Requirements Analysis. The purpose of this process is consistent with every "prepare for phase" process identified within the system development lifecycle - to assess whether the Project Team members, and the environment in which they will work, are ready for successful completion of this phase.

Description

The skills needed by the Project Team to perform System Requirements Analysis processes are dramatically different from those required to translate the requirements into a technical design. While it is certainly possible for the current team to possess the range of skills required for both phases, this assessment needs to be performed and the team profile adjusted to match the needs of System Design.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Performing Organization
- Stakeholders

Often, there is a distinct advantage to keeping as much of the original Project Team as possible while progressing from each phase to the next, thereby retaining business knowledge and functional expertise gained in prior phases. It is, however, also common for the team size to expand as the project advances into the Design phase.

The initiation of a new project phase is also the right time to assess the training needs of the existing team. Please refer to Section I, Project Planning, for a detailed approach to developing your Project Team.

It is the Project Manager's responsibility to ensure that team members have adequate equipment to perform their duties, that this equipment is configured with the proper design tools,

and that the team has access to the data repository that will be used throughout design efforts. A key activity will be the definition of the mechanisms and processes to be followed for creating and maintaining all System Design related materials, similar to the repository that was utilized for all System Requirements work products and deliverables.

This is also the time to begin to establish the environment that will likely be required when the Project Team initiates the prototyping activities.

During this phase, the Project Team's focus moves from business and functional areas to technical issues. As a result, there is often less involvement on the part of those project participants more closely aligned with the organization's functional and operational needs (the Stakeholders, Customer, Consumer, and Project Sponsor). These parties may begin to feel isolated or removed from the project due to their reduced involvement and they may not be immediately aware of much of the progress of the Project Team. While they are likely to play an active role in the discussions surrounding test planning and data conversion, they usually have limited involvement in the identification of the technical architecture and standards and the development of Technical Specifications. This situation poses a challenge for the Project Manager, since these individuals will ultimately be profoundly affected by all of these activities. The Project Manager must maintain effective communications with these individuals throughout this phase to ensure that they understand the implications of the technical decisions being made.

Business area experts, such as Customer Representatives, typically have much less involvement in System Design than in the System Requirements Analysis phase. Those areas of System Design in which they are most often involved include:

- Reviewing iterations of the prototype and user interface design.
- Defining detailed business-related algorithms that were not specified during System Requirements Analysis.
- Approving plans for converting from old system(s) to the new one.
- Validating user security schemes and authorizations.

Periodic design reviews conducted at key points during System Design often provide a way to batch user comments so that they are most useful to the software designers.

While System Design activities often reduce the demand for involvement of those participants aligned with the functional side of the organization, there is usually an opportunity for increased participation by the technical Project Team members. These team members, (often associated with Technical Services, Technical Support, ISO, and SQA), have a definite vested interest in many decisions being made and directions being set. As a result, they need to understand the extent to which the design aligns with their existing infrastructure, standards, and processes. In addition, it may be these team members who ultimately inherit, and therefore must provide long-term support for, the final system. The earlier the Project Manager can incorporate these sectors of the organization into the Project Team and make them a part of the solution, the better. The more active their involvement during System Design, the greater their buy-in to the system, and the greater their sense of ownership throughout the project.

3.2 DEFINE TECHNICAL ARCHITECTURE

Purpose

The purpose of **Define Technical Architecture** is to describe the overall technical solution in terms of the hardware platform, programming development languages, and supporting toolsets to be used in the creation and maintenance of the new system. The goal of this effort is to design a technical solution and architecture to accommodate both the initial and expected long-term requirements of the Performing Organization.

Description

The Project Team needs to understand the processing and data management capabilities, and the respective long-term strategic technical directions, of the organization that will ultimately support this application. This understanding will enable the team to determine the best approach to distributing or centralizing the data and processing capabilities of the system.

To define the technical architecture of the new system, the Project Team must perform a thorough assessment of the orga-

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Performing Organization
- Stakeholders

nization's existing infrastructure, standards, and information capabilities. Assuming that the technical platforms already in place can adequately support the new system, a design that leverages these existing platforms results in clear advantages in terms of increased productivity, decreased costs, and reduced learning curves. It is not uncommon, however, for new systems to impose technical solutions that require the extension or expansion of an organization's current architecture standards. Prime examples of this are organizations seeking to establish an Internet and Intranet presence with 24x7 accessibility, potentially introducing the necessity for new system support, security, disaster recovery, and maintenance strategies.

Issues that need to be addressed during this process include:

- Determination of the hardware and system platforms (mainframe, client/server, etc.) needed to accommodate the various Development, QA, and Acceptance environments.
- Definition of a strategy for distributing the system across the architecture (data storage and access, business logic, and user presentation layers).
- Identification of runtime and batch processing requirements, and definition of a supporting strategy.
- Assessment of reporting, archiving, and audit requirements, along with a supporting architecture.
- Determination of system interfaces and the accommodation of the supporting data.

It is during the System Design phase that the significance of the Technical Lead/Architect role increases, with responsibility to:

- establish and communicate the overall technical direction,
- ensure that design decisions made during this phase effectively support the functional requirements while leveraging existing infrastructure and standards,
- justify and defend design decisions that deviate from the existing environments,

- establish standards by which all Technical Specifications will be produced, and
- communicate with all technical support organizations (both internal as well as statewide entities).

Obviously, the Technical Lead/Architect is crucial throughout this process. Keys to the Technical Lead's success are familiarity and background with multiple technologies, and the ability to assess pros and cons of these technologies as they apply to the system at hand. As Project Manager, you need to ensure that the Technical Lead has access to additional expertise and resources, if needed.

Deliverable

- **Technical Architecture** – A document describing the overall system architecture in terms of hardware, software, tools, and peripherals, and the logical distribution of system components and processes across this architecture.

Figure 3-3 Technical Architecture Template

< Name of Agency >

Technical Architecture

< System Name >

Agency	
Project Name	
Project Sponsor	
Project Manager	
Document Date	
Prepared By	

*Enter the name of the **Agency** for which the system is being developed.*
*Enter the **Project Name**, and the names of the **Project Manager** and the **Project Sponsor**.*
*Enter the **Date** as of which this document is current.*
*Enter the names of the Project Team members by whom the document was **Prepared**.*

Figure 3-3 (Continued)

Technical Architecture

TABLE OF CONTENTS

*The **Table of Contents** should be at least two levels deep.*

1.0 DOCUMENT SCOPE

***Document Scope** describes the context and the goals of this document in a narrative.*

Example:

This document describes the Technical Architecture of the <XYZ> System that satisfies business requirements as documented in the Business Requirements Document, <Date>, and implements the functionality and satisfies technical, operational and transitional requirements described in the Functional Specification, <Date>.

The goal of this Technical Architecture is to define the technologies, products, and techniques necessary to develop and support the system, and to ensure that the system components are compatible and comply with the enterprise-wide standards and direction defined by the Agency.

This document will also:

- *Identify and explain the risks inherent in this Technical Architecture;*
- *Define baseline sizing, archiving and performance requirements;*
- *Identify the hardware and software specifications for the Development, Testing, QA and Production environments;*
- *Define procedures for both data and code migration among the environments.*

The Document Scope narrative also provides an overview of the efforts conducted to understand the existing technical environment and IT strategic direction and to determine how the system's proposed technical architecture fits into them.

Figure 3-3 (Continued)

Technical Architecture

2.0 OVERALL TECHNICAL ARCHITECTURE

2.1 System Architecture Context Diagram

*The **System Architecture Context Diagram** provides the "big picture" view of the system's architecture, and puts it in context with the rest of the Performing Organization's systems portfolio, illustrating how the system's hardware and software platforms fit into the existing environment.*

2.2 System Architecture Model

*The **System Architecture Model** represents the various architecture components that comprise the system, and shows their interrelationships.*

2.2.1 Overall Architectural Considerations

*The **Overall Architectural Considerations** section defines how additional technical requirements have been addressed by the architecture. Representative items in this section may include:*

- *Security Strategy*
- *Performance requirements*
- *Accessibility*
- *Database sizing*
- *Transaction volumes*
- *Data import and export*
- *Data encryption and decryption*
- *Disaster recovery*
- *Audit tracking*

2.3 System Architecture Component Definitions

2.3.1 System Architecture Component A

*The **Architecture Component Definitions** section provides narrative describing and explaining each architecture component in the System Architecture Model, and identifies specific elements that comprise that component in this system. The following are examples of architecture components and elements:*

Architecture Component	*Component Elements*
Database Server	*Server Hardware Configuration* *Server Operating System* *DBMS*
Client Application	*Development Tool* *Online Help Tool* *Client Characteristics*

2.3.2 System Architecture Component B

*The **System Architecture Design** section provides detailed descriptions of each product implementing architecture components, and explains the rationale for product selection.*

Figure 3-3 (Continued)

Technical Architecture

3.0 SYSTEM ARCHITECTURE DESIGN

3.1 System Architecture Component A

3.1.1 Component Functions
3.1.2 Technical Considerations
3.1.3 Selected Product(s)
3.1.4 Selection Rationale
3.1.5 Architecture Risks

*For each **System Architecture Component**, the narrative describes specific **Component Functions**, requirements and other **Technical Considerations** that were used in the decision-making process, as well as any specific **Products** selected to implement this component. The **Selection Rationale** identifies any other products that may have been considered, and provides rationale for the decision. **Architecture Risks** identifies any potential risks associated with the architecture element.*

3.2 System Architecture Component B

4.0 SYSTEM CONSTRUCTION ENVIRONMENT

*The **System Construction Environment** section details the various environments necessary to enable system construction and testing.*

4.1 Development Environment

4.1.1 Developer Workstation Configuration
4.1.2 Supporting Development Infrastructure Configuration

4.2 QA Environment

4.2.1 QA Workstation Configuration
4.2.2 Supporting QA Infrastructure Configuration

4.3 Acceptance Environment

4.3.1 Acceptance Workstation Configuration
4.3.2 Supporting Acceptance Infrastructure Configuration

*For each environment necessary for system construction (**Development**, **QA** and **Acceptance**), provide detailed specifications for the **Workstation** and **Supporting Infrastructure** that will be used (including hardware and operating system requirements, all necessary installed packages and tools, and needed directory structures that will be utilized to store all construction components).*

3.3 DEFINE SYSTEM STANDARDS

Purpose

The purpose of the **Define System Standards** process is to develop and identify programming techniques, naming conventions, and all other standards that will be used to introduce consistency and conformity throughout system development efforts.

Description

In an attempt to maximize efficiencies in the design, coding, testing and management of the system, it is important to define system standards early in the design process. System standards typically fall into three basic categories:

- Technical Development
- Configuration Management
- Release Management

Technical Development standards describe naming conventions, programming techniques, screen formatting conventions, documentation formats, and reusable components. These may be established for all projects in a large data processing/IT shop, or may be developed uniquely for a particular project. In addition, they may be unique to a development team, or industry-standard and universally accepted.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Performing Organization
- Stakeholders

Configuration Management standards provide the basis for management of the development of individual software components of the system. These standards ensure that functions such as controlling and tracking changes to the software being developed, along with backup and recovery strategies, are inherent in the development process.

Establishing Release Management standards at this point in the lifecycle ensures that the right level of planning occurs surrounding both the initial and subsequent releases of the system to the Customers and Stakeholders. These standards are also necessary for successfully managing migrations of the application to the various testing environments.

It is essential that the technical architecture system standards be firmly established before starting further System Design and System Construction activities. Deferring these activities can have a significant impact later in the project, often causing rework or discarding of completed work products. The later in the project that these decisions are made, or that prior decisions are reversed, the larger the "snowball effect" in terms of the amount of work that needs to be reviewed and potentially revised.

Deliverable

- **System Standards** – A document detailing the various standards to be applied and adhered to throughout the execution of the project. Standards applicable to each phase of the lifecycle will be identified, along with examples, where applicable.

Figure 3-4 System Standards Template

< Name of Agency >

System Standards

< System Name >

Agency	
Project Name	
Project Sponsor	
Project Manager	
Document Date	
Prepared By	

*Enter the name of the **Agency** for which the system is being developed.*
*Enter the **Project Name**, and the names of the **Project Manager** and the **Project Sponsor**.*
*Enter the **Date** as of which this document is current.*
*Enter the names of the Project Team members by whom the document was **Prepared**.*

Figure 3-4 (Continued)

System Standards

TABLE OF CONTENTS

*The **Table of Contents** should be at least two levels deep.*

1.0 DOCUMENT SCOPE

***Document Scope** describes the context and the goals of this document in a narrative.*

Example:

This document defines standards that will be followed in the development of the <XYZ> system. The following sources were considered in development of these standards:

- *<Agency> Programming Standards, <v. 1.6>*
- *<Agency> Naming Conventions, <Date>*
- *<Agency> Configuration Management Procedures, <Date>.*

All deviations from the <Agency> standards are annotated and explained.
This document addresses standards for the following areas:

- *Graphical User Interface*
- *Reporting*
- *Application Navigation*
- *Error Prevention and Correction*
- *Programming*
- *Documentation*
- *Naming Conventions*
- *Database Access and Views*
- *Data Creation and Updating*
- *Stored Procedures*

The Document Scope narrative also provides an overview of the efforts conducted to understand the existing standards in the organization, and to research those areas for which no appropriate standards exist.

Figure 3-4 (Continued)

System Standards

2.0 MODULE DEVELOPMENT STANDARDS

2.1 Graphical User Interface

Graphical User Interface *standards address such areas as:*

- *Screen Design (Resolution, layout, menus, tabs, messages, and other screen components)*
- *Visual Design (Colors, fonts, icons, buttons, symbols, system pointer, tabs and other visual components)*
- *Operational Design (GUI component behavior, operational consistency, application response (instructions and messages), and other operational behaviors)*
- *Usability Design (simplicity, clarity, fault tolerance, system feedback, and other usability attributes)*

2.2 Reporting

Reporting *standards address such areas as:*

- *Report Design (Layout, Mandatory fields, Identifying fields, Fonts, Colors, Graphical elements)*
- *Numerical Data Representation (Formats, Totals, Rounding)*
- *Usability Design (Simplicity, clarity, messages, explanations, user-controlled variables)*

2.3 Application Navigation

Application Navigation *standards address such areas as:*

- *Menu Structure (Levels, access, behavior, verbosity)*
- *Selective Navigation (Beginner, Intermediate and Expert navigation paths)*
- *Navigation Aids (Navigation buttons, keyboard commands, navigational messages)*
- *Keyboard Shortcuts (Special key combinations for frequently-used operations*

2.4 Error Prevention and Correction

Error Prevention and Correction *standards address such areas as:*

- *Input Guidance (Variable field protection, value selection, informational messages, directional messages, input fields position/presentation/behavior)*
- *Input Validation (Validation order, edit checking, alternative presentation, choice verification)*
- *Error Handling (Routines, messages, responses)*

2.5 Programming

Programming *standards address such areas as:*

- *Coding standards (Organization, structure, style, consistency)*
- *Documentation standards (Placement, verbosity, style)*
- *Development environment elements (Objects, packages, methods, variables, parameters, etc.)*
- *Debugging (Techniques, routines, messages)*

Figure 3-4 (Continued)

System Standards

2.6 Documentation

Documentation *standards address such areas as:*

- *Technical Specifications (Organization, format, level of detail, style)*
- *Test plans (Organization, format)*
- *Test results (Format, responsibility, process)*
- *Defect logs (Format, responsibility, process)*
- *User materials (Organization, presentation, format, style)*
- *Technical Documentation (Organization, presentation, format, style)*

2.7 Naming Conventions

Naming Conventions *standards address such areas as:*

- *Overall naming schema (Concepts, hierarchy, precedents)*
- *Overall naming conventions (Terminology, use of abbreviations, case, length, format, consistency, precedence, extensions, prefixes and suffixes)*
- *Development environment naming conventions by element (Module, object, package, variable, parameter, stored procedures)*

2.8 Database, Data Access and Data Views

Database *and* ***Data*** *related standards address such areas as:*

- *Database standards (Scripts, tables, rows, columns)*
- *Data access (Record locking, online updating techniques, batch updating techniques, deadlocks, navigation techniques)*
- *Data views (Creating, updating, using)*
- *Stored procedures and triggers (Conventions, techniques)*

2.9 Miscellaneous Standards

Miscellaneous *standards address any other Technical Development areas that are not covered in sections above.*

3.0 CONFIGURATION MANAGEMENT STANDARDS

3.1 Development Environment

3.1.1 Software Management

3.1.2 Database Management

Figure 3-4 (Continued)

System Standards

3.2 QA Environment

3.2.1 Software Management
3.2.2 Database Management

3.3 Acceptance Environment

3.3.1 Software Management
3.3.2 Database Management

*For each environment necessary for system construction (**Development**, **QA** and **Acceptance**), identify **Software Management** procedures (source code version control, application version control, backup and recovery, etc.) and describe procedures and controls used for **Database Management** (data sources, migration and synchronization procedures, data backup and recovery, etc.)*

4.0 RELEASE MANAGEMENT STANDARDS

4.1 Migration from Development to QA Environments

4.1.1 Software Migration
4.1.2 Data Migration

4.2 Migration from QA to Acceptance Environments

4.2.1 Software Migration
4.2.2 Data Migration

***Release Management Standards** detail how source code, compiled applications, and data will be migrated among the various environments (Development, QA, and Acceptance).*

5.0 TESTING STANDARDS

5.1 Unit Testing

5.1.1 Unit Testing Standards
5.1.2 Unit Testing Tools

5.2 Integration and System Testing

5.2.1 Integration/System Testing Standards
5.2.2 Integration/System Testing Tools

5.3 Acceptance Testing

5.3.1 Acceptance Testing Standards
5.3.2 Acceptance Testing Tools

*For each kind of testing performed (**Unit, Integration/System and Acceptance**), define **Testing Standards** and suggested approaches to setting up test cases and conducting tests, and identify and describe **Testing Tools** that should be utilized in that testing cycle.*

3.4 CREATE PHYSICAL DATABASE

Purpose

The purpose of the **Create Physical Database** process is to accommodate all of the data that needs to be managed by the system within the system database tables and files. This information must be stored in a manner that ensures its reliability, accuracy, and completeness, while minimizing redundancy and meeting system performance expectations.

Description

The Create Physical Database process expands on the Logical Data Model created during System Requirements Analysis to identify physical database schemas, file formats, and data views required by the system. While the majority of new systems developed take advantage of relational database technologies, it is important to consider the feasibility of this approach for handling the full extent of the system's data needs. Often, data will be used in the exchange of information between the system being developed and other existing legacy systems. The system interfaces may require the creation and management of data that, for valid reasons, uses other non-relational storage mechanisms.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Performing Organization
- Stakeholders

It is important to review existing database administration, data distribution, and data management policies and guidelines prior to proceeding with the definition of the physical database. These policies often dictate approaches to auditing, archiving, and recovering data that may need to be taken into consideration.

Data views are an effective way to manage the presentation of data to the user as well as to accommodate many of the security needs of the system. Sometimes, data views are overlooked until late in the project, often defined and created during the construction or testing phases in response to security or performance issues. This is a clear case of "you can pay me now or pay me more later", with the costs associated with implementing these views late in the project often exceeding what they would have been had data views been a focus in the early design efforts.

When designing the database, it is important to accurately estimate anticipated data usage and volumes. Answers to basic questions will help determine the most efficient database schemas, and will enable the team to optimize the database to achieve desired performance. Sample considerations include:

- Expectations surrounding the use of the data.
- The number of users expected to access the data simultaneously during normal usage.
- Anticipated peak user loads on the system.
- Data retention needs (e.g., is it necessary to save specific details of each record, or is it sufficient for historical purposes to simply maintain summarized data?).

Finally, it is critical to understand the data needs of all environments associated with the system being developed. Many organizations require multiple environments for development, testing, quality assurance, and production of the system, with each environment having its own unique characteristics. All of these requirements must be taken into consideration when designing and creating the physical database.

This process results in the production of database creation scripts and file utilities that, when executed, produce the physical database tables and system data files required by the system. The creation of these scripts, sometimes through the use of automated tools, equips the development team with a base upon which all future enhancements or refinements to the database can be made. Once the scripts have been modified and tested, a simple rerunning of the scripts will produce the enhanced database.

Deliverable

- **Database and System Files** – Physical data storage repositories created to support the data management needs of the application being developed, either in the form of a relational database, tables, and structures, or in the form of structured system files.

3.5 PROTOTYPE SYSTEM COMPONENTS

Purpose

The purpose of the **Prototype System Components** phase is two-fold – to provide early examples of system screens and reports that demonstrate to the Customer the proposed look and feel of the system; and to validate the applicability and feasibility of proposed technical components as they pertain to the overall technical solution.

Description

Prototyping system components is one of the most popular methods used to help the Project Team to make educated design decisions based on an actual hands-on assessment of various alternatives. Prototyping also helps to mitigate the risks associated with the introduction of new technologies or toolsets into an organization.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- Application Developers
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Performing Organization
- Stakeholders

Often, throughout the design of a system, the Project Team may be faced with having to choose from several alternative approaches. They may have to select from a variety of technical architectures, package components, or graphical user interface (GUI) designs. In order to select the best approach for the project, it is necessary to determine what best meets the Customer's needs and expectations, and is technically feasible.

Prototyping activities will often provide information on the performance and usability of the system, as well as insights into the design process.

Prototyping for Illustrative Purposes

The Functional Specification captured the operational and informational needs of the system. While there may have been some preliminary prototyping activities performed during System Requirements Analysis, these efforts are typically rudimentary, and do not contain the level of detail necessary for the design team to fully lay out the user interface aspects of the system. In order to successfully complete the design of both the user interface and the related system reports, it is often useful to conduct iterative prototypes of the system.

Aspects of the system addressed through prototyping should include screen layouts, navigation, controls (such as push buttons), and presentation styles. With respect to reports, prototyping can easily illustrate the proposed approach to queries, report selection, and filtering of data based on various reporting options.

Prototyping enables Customers to get a clearer picture of the system being developed and presents an opportunity for them to ensure that the Project Team understands their requirements. It provides all project participants with a common view of the system as it is being designed, and allows for open communications and input into how the screens and reports can be improved very early in the project. Whether these prototypes are done with paper and pen or with some elaborate design and development tool does not matter. The key is that proactively accommodating these refinements throughout the Design phase, instead of during either the development or testing of the system, eliminates many of the pitfalls that are typical of projects that exceed their allocated budget, timeframe, or both.

Another benefit to prototyping is that by actively involving the Customers in the design of the system, a sense of ownership and buy-in is created that might not otherwise be possible, or that certainly could be more difficult to achieve if the system were designed without their input.

In addition, there are advantages to engaging the Application Developers early in System Design. While these developers will make their primary contribution to the project during System Construction, involvement during System Design will enhance their overall understanding of the system, the business objectives, and the rationale behind many of the design decisions, all of which will contribute towards a stronger final product.

Prototyping as a Means of Performing Proof of Concept

In addition to being a means of illustrating and refining the design of the system, prototyping is also a valuable approach used prior to the development of the entire solution, to validate that individual components of the system will operate as expected. This can be used to confirm whether the functionality of purchased hardware or software meets expectations, for example, or to substantiate architectural decisions made earlier in System Design. Other areas of the design in which prototyping can be used include system-to-system interfaces, security, and off-the-shelf ad hoc reporting tools.

Regardless of the specific prototyping activities performed, they can serve to validate that the design solution being developed provides the best overall fit with a system that satisfies functional requirements and meets system performance goals.

A word of caution ... although the benefits of prototyping can be tremendous, and more information is learned with each iteration, prototyping without sufficient management controls can quickly take on a life of its own. It is important to set expectations from the start regarding the number of iterations that your Project Plan will accommodate. Otherwise, the Project Team can easily find itself in an endless procession of "nips and tucks" and "minor revisions" that all add up to extra effort and increased risk to the schedule and budget. Knowing that there are a finite number of prototype reviews planned can also encourage more active participation and involvement on the part of the Customers, since they should understand that they risk "missing the boat" if their feedback comes too late in the project.

In addition, many project participants often look at prototypes as an indication that the system is close to functional. This can lead to false impressions and overly optimistic time predictions if not properly managed.

Deliverable

- **Prototype** – A set of work products that 1) illustrate how the system being developed may look when finished, and serve as a model for subsequent development efforts, and 2) describe the applicability of one or more potential solutions, technologies, approaches, or components to satisfying the requirements associated with the system being developed.

3.6 PRODUCE TECHNICAL SPECIFICATIONS

Purpose

The purpose of **Produce Technical Specifications** (or 'Tech Specs') is to perform the critical process of translating all requirements captured during System Requirements Analysis into a set of documents forming the framework upon which all upcoming application development, testing, and implementation activities will be performed by the Project Team.

Description

Technical Specifications take many forms, including diagrams, structured English, decision trees, and pseudo-code. Regardless of the method used to capture and communicate the information, the main purpose is to encapsulate every possible detail needed to completely define the various technical modules that must be produced to create the new system. The accuracy and completeness of these details are essential, as these specifications drive many of the subsequent SDLC activities – most notably, the construction of each of the system modules, and the development of the test plans to be used in the eventual validation of the system.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Facilitator
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Performing Organization
- Stakeholders

During System Requirements Analysis, the role of the Business Analyst was critical to ensure that the full set of business requirements was captured accurately and completely. During System Design, the Business Analyst role expands to include the function of liaison between the business world and the technical members of the Project Team.

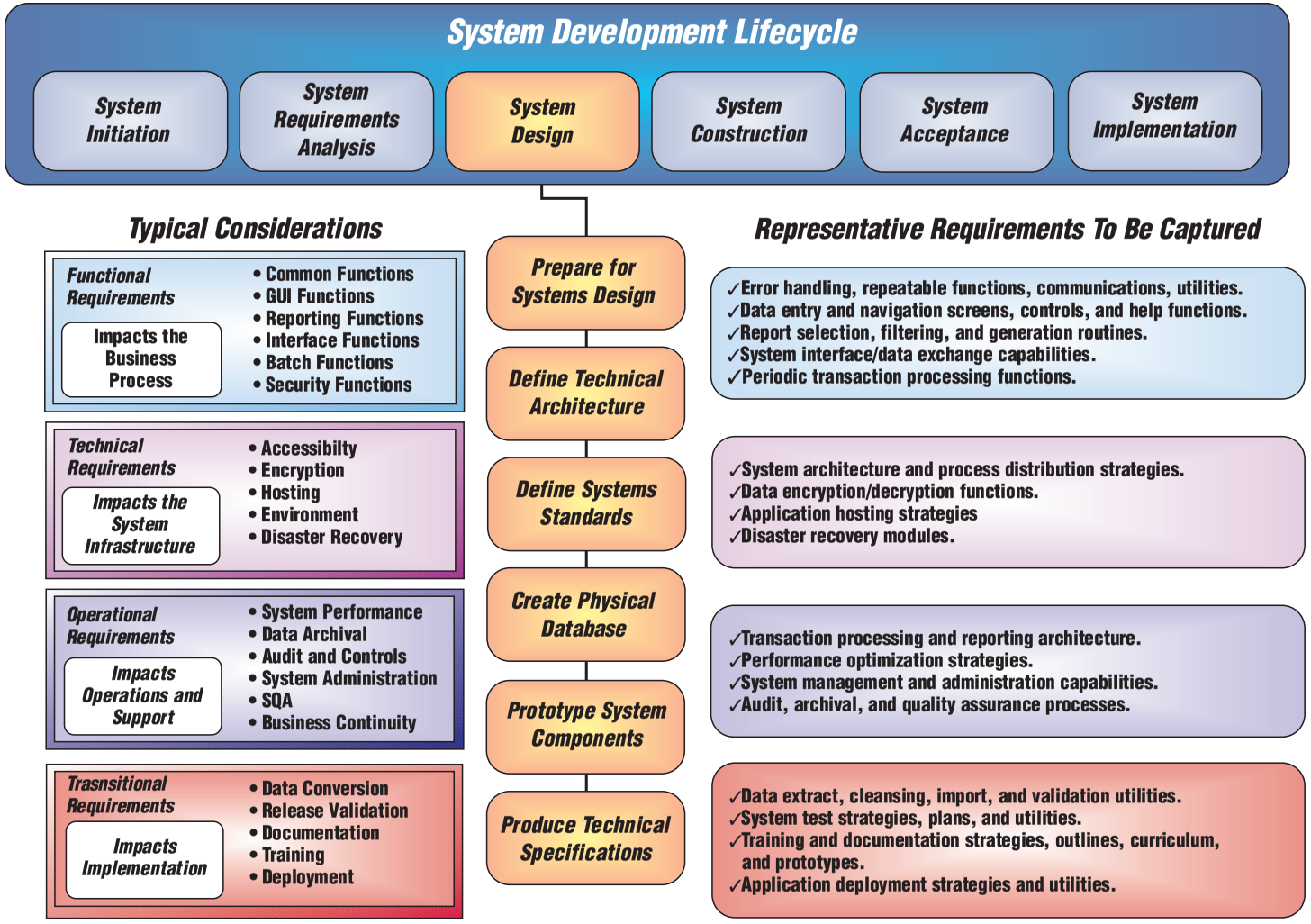
A common mistake made on many System Development projects is attempting to reduce or remove the Business Analyst function from the Project Team once the requirements have been captured. This can lead to many potential pitfalls, since it then places the burden of transforming these requirements into Technical Specifications on the shoulders of team members who may not have been actively involved in the JAD sessions or interviews from which the requirements were derived.

In fact, if project budget and staff availability permit, there can be significant benefits gained by maintaining a Business Analyst presence on the team throughout the entire project, ensuring continuity in understanding the objectives and operations of the various system components.

The Business Analyst should also have an active role in the development of both test plans and data conversion strategies. These efforts are most successful when input is solicited from the Customer Representatives, taking into consideration the full spectrum of available business expertise. Techniques used in the development of these strategies often include interviews and JAD sessions.

As with system requirements, it is important for the Project Team to address all dimensions of the solution – specifically, the functional, technical, operational, and transitional needs of the system – when producing these Technical Specifications. Figure 3-5 illustrates the types of considerations that the Project Team must keep in mind specific to System Design.

Figure 3-5 System Design Considerations



Managing the Project Team and the Customer relationship throughout the creation of the Technical Specifications can be one of the more challenging aspects of a system development project. Typically, there is pressure to produce components of the application prior to the conclusion of System Design. Reasons for this include:

- Customers begin to get anxious to "see" progress. They begin to look at the work to date as a series of documents, none of which directly results in the development of a screen or report. As a result, they begin to push for development to begin.
- The Project Team may want to begin to code those components of the system that are defined early in the design phase. Although there can be benefits to overlapping Design and Construction, the key is to do so in a controlled fashion.
- Project Team members may begin to view the production of exhaustive technical design specifications as overkill. This is particularly true if the individuals producing the specifications are those who will also develop the software. Since many of the technical details are already "in their heads", they just assume that they will recall necessary details during the development of the system.

Capturing complete, detailed Technical Specifications is essential for the development of a robust and successful system. Detailed specifications are also critical when system requirements change over time and the impact of the change on the system modules must be assessed.

As illustrated in Figure 3-5, designing a complete solution means considering each aspect of the requirements and designing a set of Technical Specifications that supports all dimensions of the project. Detailed Tech Specs not only define the specific application modules or functions to be developed, but also establish the framework for all remaining phases of the lifecycle. For example, these specifications will define:

- detailed **module specs** for all system components, whether they are common routines, GUI elements, report and batch functions, or interfaces;
- the approach for implementing the **security strategy** (defined in the Technical Architecture) throughout each module of the system;

- **system performance expectations** and a strategy for meeting them given anticipated system usage and peak processing loads;
- cumulative **testing strategies** enabling validation of all levels of the application from individual modules through a completely integrated system;
- a complete **data conversion** approach addressing the cleansing and loading of historical data as well as population of new data not currently available in any existing system;
- **documentation and training strategies** aligned with the overall application, enabling development of comprehensive training curricula, and supporting materials; and
- **deployment plans** addressing the distribution and transition of the system, that can be reviewed with and validated by the Consumers.

The Project Manager's role is to communicate the importance of producing complete and accurate Technical Specifications, demonstrate meaningful progress, and exude confidence to the Customers and Stakeholders. The best time to start delivering this message is when this process is first kicked off.

Creative and quantifiable evidence must be provided to prove that progress is being made through the development of specifications. Common indicators of progress are counts of the number of Tech Specs initiated, completed, reviewed and approved. Remember, a picture is worth a thousand words, so find ways to depict this graphically in your Project Status Reports. Everyone needs to understand the increased risks and potential cost impact of taking shortcuts, even though shortcuts could give the sense that more progress is being made because they could lead to System Construction earlier in the project.

And, don't forget to take every advantage of prototyping. This remains one of the most effective tools for generating buy-in and demonstrating progress.

Since System Construction logically follows System Design in the overall lifecycle, the team's focus will be on designing the application components needed to build the final solution. Other critical aspects of the project must also be addressed. While many of the activities surrounding these efforts may occur later in the project, the Project Manager must ensure that the planning and coordination of these efforts occur throughout System Design, coincident with the application design.

The following pages provide a detailed decomposition of three representative areas of the system – **data conversion**, **testing,** and **deployment**. The intent is to demonstrate the level of thought and consideration that must go into planning and designing every element outlined in the Technical Specification.

Data Conversion

Much of the data required by the system will be entered through normal day-to-day business operations, whether through manual input or through automated mechanisms. Additional data may be required, however, before the system can effectively begin operation, such as:

- **Historical data,** typically found on existing legacy systems, that may need to be migrated to the new system to provide a basis for future calculations, historical or trend reports, etc.
- **Reference data,** also known as lookup data, which can be used to populate common tables upon which other data is dependent (e.g., system codes that might be referenced across multiple tables). This information may or may not reside on existing systems, depending upon how the eventual design of the new system maps to the legacy environments.
- **New data,** essential to the initial operation of the system being built, that may not be available on any legacy systems.

Whether or not the data the new system requires exists on legacy systems (or in spreadsheets, or on scraps of paper, etc.), the Project Manager must ensure that the Project Schedule includes the steps needed to obtain all required data in a format compatible with the new environment. This often necessitates the development of conversion and migration software modules, to support and ensure successful completion of the data conversion. Research may also be needed to determine whether data is valid, and cooperation between multiple organizations may be required as attempts are made to identify and resolve conflicting data.

It is not uncommon for many Project Managers to delay planning of this effort until later in the project. As a result, there is often a last minute scramble to try to account for and address all of the potential issues that surround the collection and val-

idation of the data. By evaluating data needs and planning how best to address them during System Design, the Project Manager will better position the Project Team to take all necessary actions throughout the construction of the system.

While evaluating initial data needs (often referred to as "Day One" data, since it refers to the data needed on the first day of system operation), it is important to perform a gap analysis against all existing data. This identifies data that is needed but not immediately available. This process may also require the development of software modules, whose sole purpose is to pre-populate tables as part of the overall data conversion process, but which will likely not be used again once the system is operational.

Ultimately, the goal is to develop a data conversion plan. The plan outlines all of the necessary data population efforts required prior to the system going live, and assigns responsibilities for these efforts to the Project Team, Customers, and anyone else who may need to take an active role in this effort.

Because data is often only as good as the source from which it originated, you need to ensure that you involve your Customers in evaluating and validating the information that may eventually be loaded into your system. Often there are historical implications or nuances embedded in the information that may not immediately be evident to someone unfamiliar with the data. The data itself may also imply business rules that may not have been captured during System Requirements Analysis. Historical data often contains "dummy" or otherwise invalid data to flag the existence of an exception situation. Without planning for direct and active involvement of your Customers during the data conversion process, the risk of missing or mishandling critical system data is greatly increased.

One final consideration when planning data conversion is its potential impact on an organization's operations. The time it takes to physically convert, populate, and validate system data can be lengthy, depending upon the volume of data, and the variety of sources from which information will be pulled. It is not uncommon for data conversion plans to require that legacy data be "frozen" as of a certain date, after which any data entered into the legacy system will not automatically be converted. The data conversion plan must ultimately account for how any data entered beyond the freeze date will be entered

into the new system. If there is a need to run the new and legacy systems in parallel for some period of time to allow for validation of the new system, there may be additional data conversion implications that must be addressed. All identified impacts should be captured in the Project Implementation and Transition Plan and the Organizational Change Management Plan, both defined in the Project Planning phase of the Project Management Lifecycle.

Testing

Test plans created in the Produce Technical Specifications process define the overall strategy for validating the functionality of the system being developed, as well as the individual test cases that will be performed in the execution of this strategy. Additionally, the environments in which these tests will be executed must be defined in detail.

Four common types of testing are:

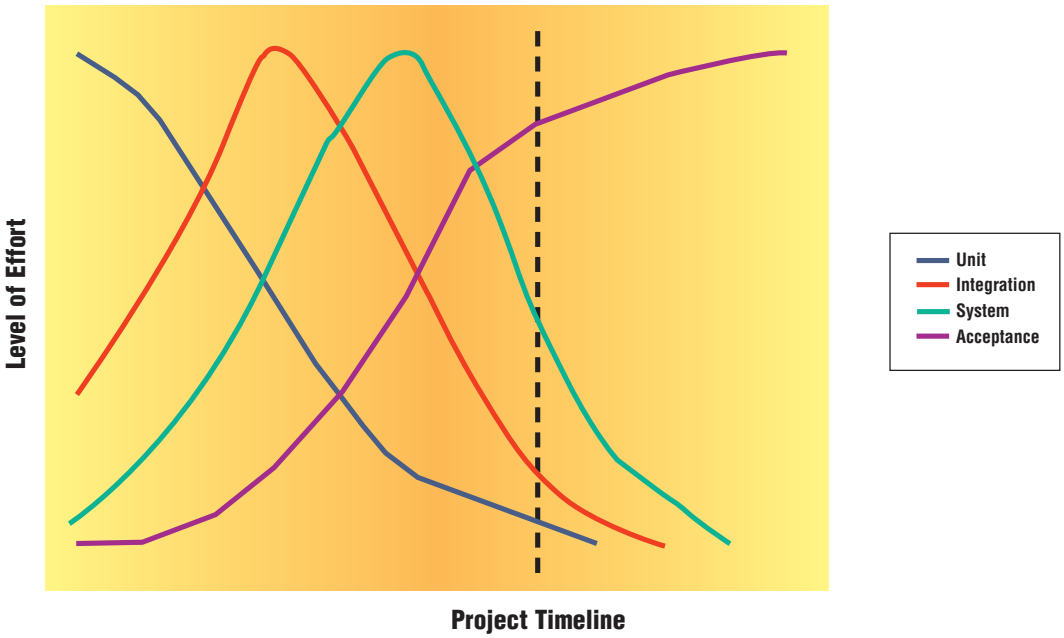
- **Unit Testing,** where individual system components are independently tested as they are developed to ensure that each logic path contained within each module performs as expected. Many tests performed during unit testing can be used for more than one module (error handling, spell checking of screens and reports, etc.).
- **Integration Testing,** where multiple, related elements of the system are tested together to validate components of the system, and to ensure that the appropriate edits and controls are functioning correctly. This testing concludes with the entire system being tested as a whole. "Bottom up" and/or "top down" testing approaches can be used. With bottom up testing, the lowest level modules are created and tested first, and successive layers of functionality are added as they are developed. Top down testing takes the opposite approach, where the highest-level modules are developed and tested, while lower level "stubs" are created and invoked until the actual modules are available. These stubs are temporary software modules that are created in order to enable the higher-level routines to be validated, but that do not yet perform the full set of functions needed by the system. Most testing strategies employ a mix of both approaches.

- **System Testing,** where the entire system is linked together and tested to validate that it meets the operational requirements defined during System Requirements Analysis. Factors that are commonly tested at this level include performance, load, boundary, and external interfaces.
- **Acceptance Testing,** where the Customer Representatives, and potentially Consumers and Stakeholders, perform validation tests to ensure that the developed system meets their expectations and needs. The results of this testing usually determine whether or not the system is ready to be released into production, so it is critical to define and understand the plan for completing this testing as early in the project as possible.

Thoroughly documented and detailed test cases provide two advantages. First, they enable the execution of these tests to be performed by any Project Team member, not just those team members that created the test cases. Secondly, they provide the basis for future regression testing efforts, where all aspects of system functionality are revalidated whenever changes are introduced to the system (most likely during the long-term maintenance and support of the system once it is in production). Involving the SQA Analyst in the development or review of these test cases can help to ensure that they can be leveraged by both the initial Project Team and the Performing Organization once they have assumed ownership of the system.

The following chart illustrates when each of these testing cycles is used within the overall testing lifecycle. There is a heavy emphasis on unit testing at the beginning of development efforts, with a gradual shift through Integration and System Testing, and finally into User Acceptance Testing efforts as all elements of the system become available.

Figure 3-6 Typical Testing Patterns



As Figure 3-6 illustrates, testing cycles overlap each other, primarily because multiple components of the system are in different stages of the lifecycle at any point in time. As illustrated by the dashed line, it is possible that on any given day, the system could be undergoing all stages of testing concurrently. Reasons for this include:

- Modules that may have been developed early in System Construction may already have gone through unit and system testing, while unit testing of recently developed modules may just be starting.
- Testing may uncover problems with the system, frequently resulting in coding changes being made to one or more modules, which then require retesting.

Since testing is often iterative and testing activities may occur concurrently, it is important to ensure that the testing strategy accommodates these scenarios. Performing test planning activities at this stage of the project is critical so that testing participants are fully aware of the time commitments required to prepare for and execute these tests. When developing a test

plan, it is important to assess and define many aspects of the testing strategy. Factors to consider include:

- Testing objectives
- Scope of testing (both what is in and what is out of scope)
- Responsibilities
- Testing approach
- Test data needed to execute the tests
- Criteria for suspending and resuming testing
- Testing sequence
- Defect reporting and criteria

Often one of the most difficult aspects of testing an application is defining and creating the appropriate set of test data needed to validate system functionality. This is especially true in environments that require special processing of data at the end of specific time periods (monthly, quarterly, annually, etc.), or need to manage data across multiple fiscal years. Preparation of this data can be very time consuming, and it is in System Design that the scope and responsibilities for data preparation must be clearly defined.

Also, while the creation of representative or "dummy" test data may be acceptable for tests performed internally by the Application Developers, real or meaningful data should employed in any testing that involves Customer Representatives.

Deployment Planning

By this point, the Project Manager and Project Team have determined what needs to be built, how to build it, and how the Performing Organization is going to use it. The one remaining piece of the puzzle is to identify how the system being created is going to be made available for use once testing has been completed.

Again, the tendency might be to delay this planning since the actual deployment of the system may be far in the future. Proper planning of deployment activities, however, is the key to their success. The method through which the system will be made available may dictate the need for certain System Construction activities, and the testing process created during System Design must be robust enough to validate this deploy-

ment process. The full Project Team, therefore, must understand the deployment plan and its effect on other design, development, and testing activities. This information must also be clearly communicated to, and validated with, the Customers to ensure compatibility with their existing operations, infrastructure and expectations. Factors such as whether the system will require new production hardware must be identified and confirmed early in the project, to allow for proper planning of fiscal and logistical impacts (see corresponding Project Budget and Project Implementation and Transition Plan templates in the Project Planning phase of the Project Management Lifecycle.)

When the time comes to move the application into production, there will be a set of activities for the team to perform. To the extent possible, this set of activities should be tested before actual deployment to ensure that the deployment will proceed smoothly.

A final element of the deployment strategy is the communication of the overall plan. This will allow for coordination with the data conversion plan (ensuring availability of data), and will enable the Consumers to coordinate the system rollout with their day-to-day business activities, generating enthusiasm and ownership toward the system as they prepare for its arrival. This topic is explored in more detail in the Project Planning phase of the Project Management Lifecycle.

Deliverable

- **Technical Specifications** – A compilation of system diagrams, module specifications, and test plans that serve as a detailed, comprehensive blueprint for the system.

Figure 3-7 Technical Specifications Template

< Name of Agency >

Technical Specifications

< System Name >

Agency	
Project Name	
Project Sponsor	
Project Manager	
Document Date	
Prepared By	

*Enter the name of the **Agency** for which the system is being developed.*
*Enter the **Project Name**, and the names of the **Project Manager** and the **Project Sponsor**.*
*Enter the **Date** as of which this document is current.*
*Enter the names of the Project Team members by whom the document was **Prepared**.*

Figure 3-7 (Continued)

Technical Specifications

TABLE OF CONTENTS

*The **Table of Contents** should be at least two levels deep.*

1.0 DOCUMENT SCOPE

***Document Scope** describes the context and the goals of this document in a narrative.*

Example:

This document describes the Technical Specifications or the <XYZ> System that will be developed to satisfy business requirements and implement functionality as documented in the Business Requirements Document, <Date>, and the Functional Specification, <Date> and confirmed via <XYZ> System Prototype, accepted on <Date>.

The goal of this Technical Specifications document is to define the system and its development and testing strategies in enough detail to enable the Application Developers to construct and test the system with minimal need for further explanation.

This document places the system in its context from Technical Architecture, Customer Interface, and System Development perspectives, provides detailed Module Specifications for all its components, details Unit, Integration and System Plans, and outlines Deployment and Transition plans.

Figure 3-7 (Continued)

Technical Specifications

2.0 SYSTEM ARCHITECTURE

System Architecture *section provides the "big picture" view of the system from technical, functional and development perspectives, and puts it in context with the rest of the organization's systems portfolio. This section repeats some of the information from Section 2 (Overall Technical Architecture) of the Technical Architecture document, and from Section 2 (General Functional Specifications) of the Functional Specification document, with their contents refined as a result of the prototyping and other System Design efforts.*

2.1 Refined System Context Diagram

The ***Refined System Context Diagram*** *shows how this system integrates into the Agency's application portfolio. All external dependencies and influences should be noted, as well as all data sources and outputs.*

2.2 Refined System Architecture Context Diagram

The ***Refined System Architecture Context Diagram*** *shows how the system's hardware/software platform fits into the existing environment.*

2.3 Refined System Architecture Model

The ***Refined System Architecture Model*** *represents the various architecture components that comprise the System, and shows their interrelationships. This model presents the view of the system from the technical architecture perspective, as opposed to the Consumer-driven perspective of the System Interface Diagram.*

2.4 Refined Business Flow Diagram

The ***Refined Business Flow Diagram*** *shows how the Customer and Consumer business units will interface with the system.*

2.5 Refined System Interface Diagram

The ***Refined System Interface Diagram*** *shows the application structure (menu structure and navigation of the online application) and batch/utility structure (organization and flow of reporting and other interfaces), which is a refinement of the System Interface Diagram from the Functional Specification document. The System Interface Diagram presents the view of the system from the Consumer perspective.*

Figure 3-7 (Continued)

Technical Specifications

2.6 System Development Diagram

*The **System Development Diagram** shows the system from the development perspective, according to how the components of the system will be coded, tested, and integrated. Beginning with the application structure established in the Functional Specification, a modified list of sub-systems will be identified based on development efficiencies and testing and deployment considerations. Each sub-system consists of a number of Modules, which will be assigned to individual Application Developers for coding and unit-testing; each sub-system constitutes a work packet that will be assigned to a group of Application Developers for construction and integration testing.*

3.0 MODULE SPECIFICATIONS

***Module Specifications** detail the design of each system module, organized by sub-system (following the System Development Diagram). A module may be a collection of code that will present a Consumer interface, or it could be a utility, a database stored procedure, or another common object.*

3.1 Sub-System A

Depending upon how the system has been decomposed into sub-systems, these Sub-System sections contain specifications for all Modules comprising the sub-systems. Sub-systems may be defined by functional area (e.g., security, reports, etc.), or by business focus (e.g., accounts receivable, payroll, etc.)

3.1.1 Module A-1

*A **Module** may be a collection of code that will present a Consumer interface, or it could be a utility, a database stored procedure, or another common object. Each module is described in sufficient detail as to enable the Application Developers to construct and test it with minimal further explanation:*

3.1.1.1 Module Overview

***Module Overview** explains how the module will satisfy the business requirements, and how it fits into the sub-system.*

3.1.1.2 Interface Prototype

***Interface Prototype** shows the module's interface (if applicable), as accepted by the Customers during the Prototype System Components process.*

3.1.1.3 Customer Decision-Maker(s)

***Customer Decision-Makers** lists the Customers who have the sign-off authority to accept the module.*

3.1.1.4 Customer Representative(s)

***Customer Representatives** lists the Customers who are the Functional Subject Matter Experts for this module (They will answer questions about the module and will conduct its acceptance testing.)*

Figure 3-7 (Continued)

Technical Specifications

3.1.1.5 Business Requirement(s)

Business Requirements *provides a tie-back to the Business Requirements Document.*

3.1.1.6 Inputs

Inputs *details all data sources, Consumer input, etc. that will provide data to the module.*

3.1.1.7 Interfaces

Interfaces *details how the Consumers will interact with the module's interface components, and how those components will behave in all circumstances.*

3.1.1.8 Security Considerations

Security Considerations *identify how the security strategy will be implemented in the module.*

3.1.1.9 Logic Flow

Logic Flow *details how the business rules will be implemented by module code.*

3.1.1.10 Outputs

Outputs *details all data stores, displays, etc. created or modified as a result of the module's execution.*

3.1.1.11 Database Access

Database Access *explains how the module will navigate the database(s) to obtain, update, create or delete the data, and which data sources, tables, records, etc. will be used in the process.*

3.1.1.12 Common Elements Used

Common Elements Used *lists all the common objects, stored procedures, etc. that will be used by the module.*

3.1.1.13 Module Review Process

Module Review Process *outlines what QA procedures will be used to review the module and its test results.*

3.1.1.14 Audit Tracking

Audit Tracking *details how updates to data and access to or utilization of system components will be recorded and tracked for auditing purposes.*

3.1.1.15 Special Considerations

Special Considerations *allows compilation of all other Requirement, Design and/or Construction considerations that the Application Developer should be cognizant of.*

3.1.1.16 Unit Test Plan

Unit Test Plan *details how the module will be tested, once developed.*

Figure 3-7 (Continued)

Technical Specifications

Unit Test Case
- Unit Test Case Number:
- Unit Test Case Name:
- Purpose of Test Case:
- Unit Test Data:
  - Data Source A — Value(s):
  - Data Source B — Value(s):
- Navigation:
  - Navigation Directions
- Expected Results:
  - Narrative
- Comments:
  - Additional Testing Consideration

Unit Test Results:
- Tester:
  - Name
  - Date — Time
- Results:
  - Passed: _____ — Failed: _____
  - Justification:

Unit Test Case Number *allows quick reference to test case; should be based on module identification.*
Unit Test Case Name *provides a brief description of the condition/scenario being tested.*
Purpose of Test Case *identifies those functions that the test is intended to validate.*
Unit Test Data *identifies data values (or conditions) that need to be set in order to conduct the test case.*
Navigation *provides a sequence of activities that need to be performed to set up and execute the test.*
Expected Results *provides a comprehensive description of how the module is expected to react to the test case, and/or what data values (or conditions) are expected as a result of the test.*
Comments *provides additional considerations for the test (expected Fail conditions, etc.)*
Unit Test Results *allows the tester to record the results of the unit test.*
Tester *enters his/her* ***Name****, and* ***Date*** *and* ***Time*** *of the test.*
Tester certifies the test as ***Passed*** *or* ***Failed****, and provides a* ***Justification*** *for that certification. In the event of a failure, and depending upon how defects are being captured and tracked, this justification may be a description of the problem encountered, or may simply contain a reference to the defect log, where a detailed description of the error would be maintained.*

3.1.2 Module A-2

3.2 Sub-System B

3.2.1 Module B-1

3.2.2 Module B-2

Figure 3-7 (Continued)

Technical Specifications

4.0 INTEGRATION TEST PLAN

Integration Test Plan *details the activities to be performed in integration testing.*

Sub-system modules are organized into ***Integration Packets*** *to facilitate integration testing. The same module (or a series of modules) can be included in different, smaller or larger, Integration Packets depending on the aspects of the system being integrated and tested.*

4.1 Integration Packet 1

Integration Test Case
Integration Test Case Number:
Integration Test Case Name:
Module List:
Purpose of Integration Test Case:
Integration Test Data:
Data Source A Value(s):
Data Source B Value(s):
Navigation:
Navigation Directions
Expected Results:
Narrative
Comments:
Additional Testing Consideration
Integration Test Results:
Tester:
Name:
Date: Time:
Results:
Passed: _____ Failed: _____
Justification:
Verifier:
Name:
Date: Time:

Figure 3-7 (Continued)

Technical Specifications

Integration Test Case Number *allows quick reference to test case; should be based on module identification.*
Integration Test Case Name/Purpose *provide a brief description of the scenario being tested.*
Module List *identifies system modules included in the Packet*
Integration Test Data *identifies data values (or conditions) that need to be set in order to conduct the test case.*
Navigation *provides a sequence of activities that need to be performed to set up and execute the test.*
Expected Results *provides a comprehensive description of how the Packet is expected to react to the test case, and/or what data values (or conditions) are expected as a result of the test.*
Comments *provides additional considerations for the test (expected Fail conditions, etc.)*
Integration Test Results *allows the tester to record the results of the test.*
Tester *enters his/her* ***Name****, and* ***Date*** *and* ***Time*** *of the test, certifies the test as* ***Passed*** *or* ***Failed****, and provides a Justification for that certification. As with Unit testing, this justification may contain descriptive text, or may refer to an entry in the project's defect log.*
Verifier *verifies that the Integration Test was conducted as described, and produced reported results.*

4.2 Integration Packet 2

5.0 SYSTEM TEST PLAN

System Test Plan *details the activities to be performed in integration and system testing.*

Sub-systems and system modules are organized into ***System Test Packets*** *to facilitate system testing. The same packet, or the system as whole, may be tested numerous times to verify different aspects of its operation.*

5.1 System Test Packet 1

System Test Case
System Test Case Number:
System Test Case Name:
Module List:
Purpose of System Test Case:
System Test Data:
Data Source A Value(s):
Data Source B Value(s):
Navigation:
Navigation Directions
Expected Results:
Narrative
Comments:
Additional Testing Consideration

Figure 3-7 (Continued)

Technical Specifications

System Test Results:
Tester:
Name:
Date: Time:
Results:
Passed: _____ Failed: _____
Justification:
Verifier:
Name:
Date: Time:

System Test Case Number *allows quick reference to test case; should be based on module identification.*
System Test Case Name/Purpose *provide a brief description of the scenario being tested.*
Module List *identifies system modules included in the Packet*
System Test Data *identifies data values (or conditions) that need to be set in order to conduct the test case.*
Navigation *provides a sequence of activities that need to be performed to set up and execute the test.*
Expected Results *provides a comprehensive description of how the Packet is expected to react to the test case, and/or what data values (or conditions) are expected as a result of the test.*
Comments *provides additional considerations for the test (expected Fail conditions, etc.)*
System Test Results *allows the tester to record the results of the test.*
Tester *enters his/her* ***Name****, and* ***Date*** *and* ***Time*** *of the test, certifies the test as* ***Passed*** *or* ***Failed****, and provides a Justification for that certification. In the event of a failure, and depending upon how defects are being captured and tracked, this justification may be a description of the problem encountered, or may simply contain a reference to the defect log, where a detailed description of the error would be maintained.*
Verifier *verifies that the System Test was conducted as described, and produced reported results.*

5.2 System Test Packet 2

6.0 ACCEPTANCE TEST PLAN

Acceptance Test Plan *details the activities to be performed in integration and system testing.*

Modules, groups of modules and sub-systems are organized into ***Acceptance Test Packets*** *to facilitate Customer Representative testing of the system.*

6.1 Acceptance Test Packet 1

Acceptance Test Case
Acceptance Test Case Number:
Acceptance Test Case Name:
Module List:
Purpose of Acceptance Test Case:

Figure 3-7 (Continued)

Technical Specifications

Acceptance Test Data Preparation:
Data Preparer:
Data Sources and Values:
Acceptance Case Description:
Business Rules, Requirements and Conditions being tested:
Navigation directions:
Expected Results:
Narrative
Comments:
Additional Testing Consideration
Acceptance Test Results:
Tester:
Name:
Date: Time:
Results:
Passed: _____ Failed: _____
Justification:
Defect Resolution:
Application Developer:
Resolved Date:
Re-Tester:
Name:
Date: Time:
Results:
Passed: _____ Failed: _____
Justification:
Approval:
Name:
Date: Time:

Acceptance Test Case Number *allows quick reference to test case; should be based on module identification.*
Acceptance Test Case Name/Purpose *provide a brief description of the condition/scenario being tested.*
Module List *identifies system modules included in the Packet*
Acceptance Test Data Preparation *describes how the Data Preparer will prime Data Sources with Values that will provide realistic and understandable test scenarios for Customer Representatives.*
Navigation Directions *provide a guide for the Customer Representative testing the Packet on a proper sequence of activities to set up and execute the test.*
Expected Results *provides a comprehensive description of how the Packet is expected to react to the test case, and/or what data values (or conditions) are expected as a result of the test.*

Figure 3-7 (Continued)

Technical Specifications

Comments *provides additional considerations for the test (expected Fail conditions, etc.)*
Acceptance Test Results *allows the tester(s) to record the results of the test.*
Tester
In case of a ***Defect****, the Packet is passed to an* ***Application Developer*** *for* ***Resolution****; the* ***Date*** *of resolution is recorded, and the Packet is passed back for further Acceptance Testing.* ***Re-Tested*** *enters his/her* ***Name****, and* ***Date*** *and* ***Time*** *of the test, certifies the test as* ***Passed*** *or* ***Failed****, and provides a* ***Justification*** *for that certification. In the event of a failure, and depending upon how defects are being captured and tracked, this justification may be a description of the problem encountered, or may simply contain a reference to the defect log, where a detailed description of the error would be maintained.*

A Customer Decision-Maker (or Representative) approves the test results by entering his/her ***Name****,* ***Date*** *and* ***Time*** *of the* ***Approval****.*

6.2 Acceptance Test Packet 2

7.0 DEPLOYMENT AND TRANSITION PLANS

The ***Deployment and Transition Plans*** *section outlines the activities to be performed during System Implementation. The details identified in this plan form a subset of the overall Project Implementation and Transition Plan, defined in the Project Planning phase of the Project Management Lifecycle.*

7.1 Consumer Training and Deployment
7.2 Data Preparation
7.3 Software Migration
7.4 Production Start-up
7.5 Production Verification
7.6 Performing Organization Training and Transition

Consumer Training and Deployment *deals with training and preparing Consumers for system deployment.*
Data Preparation *deals with plans for data conversion, data cleansing, and data migration in preparation for system deployment.*
Software Migration *outlines an approach for migrating developed software to Production, and making it available to Consumers.*
Production Start-up *considers all other (outside data preparation and software migration) activities necessary to prepare and start up the System in Production.*
Production Verification *deals with all the tasks that need to be performed to make sure the version of the System migrated to Production is functioning properly.*
Performing Organization Training and Transition *outlines plans for training and turning over system support responsibilities to the Performing Organization.*

8.0 OPERATIONAL CONSIDERATIONS

A high-level description of how the technical architecture supports and addresses the ***Operational*** *needs of the system is presented in this section. Items include load and update procedures, report production and distribution, data archival and retrieval, backup and recovery, periodic and on-demand procedures, incident reporting, Consumer support requirements, enhancement request processing, etc.*

Measurements of Success

The immediate measurement of success for System Design is the acceptance of all deliverables by the Customer, while the eventual measurement is whether or not the system can be developed according to plan.

Meanwhile, the Project Manager can still assess how successfully the project is proceeding by utilizing the measurement criteria outlined below. More than one "No" answer indicates a serious risk to the eventual success of the project.

Figure 3-8

Process	Measurements of Success	Yes	No
Prepare for System Design	Do all team members have experience with (or training on) the tools that will be used in this phase?		
	Is the team comfortable with the process defined for managing the deliverable repository?		
Define Technical Architecture	Has the proposed architecture been reviewed by an independent third-party subject matter expert?		
	Do your Customers understand the potential impact that the proposed architecture may have on their operations, and agree that the defined architecture supports both their immediate and long-term needs?		
Define System Standards	Have the technical and configuration management standards been reviewed and approved by an agency's SQA Administrator or equivalent?		
	Have standards been defined and accepted that address the strategy for managing future releases of the system?		
Create Physical Database	Were the Performing Organization's data administration policies and standards considered in creating the database?		
	Was the database created using scripts from an automated tool to ensure consistency, repeatability, and maintainability of future builds of the database?		
	Has an independent third-party subject matter expert reviewed the physical database design?		
Prototype System Components	Has the Customer been involved in defining which aspects of the system would be prototyped and reviewed?		

Process	Measurements of Success	Yes	No
Prototype System Components (Continued)	Has Customer feedback been incorporated into the prototype?		
	Have proof of concept activities been performed for all aspects of the system believed to pose the greatest risk (i.e., those components of the system that are new to the environment, that do not adhere to existing standards, or that introduce architectures not currently validated in the environment)?		
Produce Technical Specifications	Has a gap analysis been performed between the system components identified in the Functional Specification and the system modules defined in the Tech Specs?		
	Have the Customers been involved throughout System Design in the review of the Tech Specs to ensure that they have both a sense of the progress being made, as well as confidence in the final design?		
	Is the Customer satisfied that the solution not only addresses the security requirements, but also adheres to existing organizational practices?		

Phase Risks / Ways to Avoid Pitfalls

PITFALL #1 – HOW DID BUILDING MY DREAM HOME TURN INTO SUCH A NIGHTMARE, OR HOW COULD I POSSIBLY HAVE EXCEEDED MY BUDGET BY THAT MUCH?

Everyone has a vision of a dream home, whether it is a cottage in the Adirondacks, a villa on the shores of the mighty Hudson, or a mansion that backs up to the 18th green at your favorite country club.

Building that dream home, however, can quickly turn into a nightmare if not approached correctly. The Project Team – architects, contractors, electricians, interior designers, inspectors, in addition to you and your loved ones – can quickly outnumber ants at most picnics. To complicate things even further, decisions made at every phase of the building process set the stage for the work that remains.

- The layout of your property may influence the style of house that you choose and the location and number of entrances

- The style of the house that you are building determines how many floors it will have and how it will be framed
- The design of each floor enables you to determine the exact layout of the rooms and how to run the wiring and plumbing
- How the rooms flow together lets you choose coordinated color schemes and common decorating themes
- And on and on …

The one thing that pulls it all together – the key to putting the "dream" in your dream home – is that clear, shared vision. Your dream home's worst enemies? ***Indecisiveness*** and ***impatience***.

Indecisiveness, (or the "but it's just a little change" syndrome), results in your "vision" being more like a blur. Decisions made early in the process establish the blueprint for your home, and set the Project Team in motion. The foundation is poured, the framing goes up, the plumbing is plumbed, etc. When after-the-fact decisions are introduced ("You know, we've thought about it, and we'd really like the kitchen moved three feet to the left.") there will be a ripple effect on the rest of the construction.

Impatience is that strong desire to see something happening, even though your vision is still being formulated – to pour that first load of cement, or to drive that first nail. This is analogous to putting the cart in front of the horse, if for no other reason than to convince yourself that there really is a cart and a horse. This does provide the illusion of quick results, but also often results with the wrong kind of cart, and an animal hooked up to it that, at best, looks somewhat like a horse … maybe … from a distance. Invariably, you're going to have to go back and redo some of what you had hoped was already behind you.

These factors combined can have the same effect on your budget as fast food has on your cholesterol. In other words … it's gonna go up, and it's gonna go up fast.

But how does this relate to building your new dream system?

Much like building a new home, building a new system requires that the foundation upon which it is to be built be established, "locked down", and communicated as early in the project as possible. That's the whole purpose of **Define Technical**

Architecture and **Define System Standards**, early in the System Design phase. And just like with the house, changes to the system architecture late in the project ***(indecisiveness)*** and jumping into module design or System Construction ahead of completing the architecture ***(impatience)***, can have a dramatic impact. That's why it's necessary to define your architecture and standards first, and to make sure they are as complete and as accurate as possible.

PITFALL #2 – UMM ..... WHAT?!?

Marcelle Marceau, the famous French mime, made a career out of mastering the unwritten and unspoken word. His ability to represent ideas or concepts and even to tell complete stories through only his actions entertained audiences around the world. Fortunately for Mr. Marceau, there was very little impact if his rendition of "Cooking a Gourmet Dinner" was mistakenly interpreted by an audience member as "Mailing My Shih Tzu to Schenectady."

The same obviously cannot be said for developing detailed Technical Specifications. Misinterpreting the requirements of a certain function, or the design of a specific module, can have significant impact on your project. This is further complicated due to the fact that these differences in interpretation are often not uncovered until well into the testing phases of the project, at which time the actions needed to correct the problems can be enormous.

That is why, when producing Technical Specifications, the appropriate level of detail is everything. Don't assume that the intended audience (your Application Developers) has the same level of understanding and expertise as your Analysts and Technical Leads have. For one thing, it is very likely that they have not been intimately involved in the requirements gathering or design activities. Therefore, they also do not have the same level of familiarity with the Customer, Stakeholder, or Consumer, or more specifically, with their business needs, as do those Project Team members that have been interacting with them regularly since the project started.

As a result, all of the information and details must be captured and communicated in the Tech Specs. You cannot assume that those individuals reading these specifications can correctly fill in the gaps, or that they'll even realize that there is a gap that requires filling!

So remember, when it comes to producing and validating your Technical Specifications, it pays to spend the time capturing the details up front, before you (and your Shih Tzu) find your Developers saying, "Return to Sender."

PITFALL #3 – HEY, BUDDY, CAN YOU SPARE SOME HARDWARE?

You've spent countless hours designing the best system ever devised. You've checked and double-checked to ensure that each and every requirement has been addressed, and that every data item is accounted for (your CRUD matrix is the talk of the town!). Not only that, but your team also managed to build the system in less time than was originally estimated, and you're already thinking about how sweet it will be to park in the Employee of the Month parking spot. All that remains is the testing.

And that's when the walls start to crumble. It seems that while you convinced yourself that you had devised a clever testing strategy, you were so focused on making sure that you could validate the system functionality that you may have overlooked one little detail … the hardware needed to support all of the testing environments. And now that you're neck deep in people looking for results, you've got to explain why testing will be delayed and why you've got to now find a way to obtain (translation – beg, borrow, steal, BUY?) the hardware to support integration testing. And user acceptance testing. And QA testing. Suffice it to say, there goes the parking spot.

Of course, this could all have been avoided by looking at the full testing picture. This includes not only defining how you plan to confirm that the system performs to expectations, but also that you've got the hardware, data, and all other resources needed to execute the tests. The time to think about this is now, during System Design.

Frequently Asked Questions

When is prototyping complete?

There is no absolute answer to this question. Theoretically, prototyping is complete when all Customer feedback has been received, reviewed, and accommodated for in the design. More realistically, the Project Manager will need to find a balance between the benefits of yet one more iteration and the associated cost to the project's budget and schedule. By clearly stating the expectations up front (i.e., the number of iterations accommodated for in the Project Schedule), the stage will be set to alter this approach up or down through the standard change management process being deployed on this project. Ultimately, the real question becomes, "Is the juice worth the squeeze?" - or stated another way, "At what cost is your Customer willing to pursue perfection?"

If you determine that the Customer environment in which you are working has a strong preference for an iterative prototype and design approach, you may want to consider choosing a methodology that supports iterative development (such as RAD). These methodologies allow for cycles of requirements definition throughout the development process, and are most typically used in rapidly changing Customer environments, or on projects that require quick delivery. To be successful, these approaches also are very dependent upon close Customer interaction throughout the prototype revisions.

How can the Project Team determine that all security concerns have been addressed?

Unfortunately, there is no shortcut available, and no substitute for an exhaustive review of the system's data elements, and the management of security around each element. Creation and review of a CRUD matrix, clearly delineating Consumers with authority to Create, Read, Update, and Delete system data, is one of the more traditional techniques used to ensure that the full range of security needs has been addressed. By understanding the processes through which these operations are performed, and mapping these processes to the Consumers authorized to execute them, a high degree of confidence in the security solution can be achieved.

Shouldn't definition of the technical architecture precede business requirements gathering? Don't I need to know what flashy new technology I can promise my Customers?

You must be the one who bought a backhoe to put up a birdhouse! Just as it's wise to have a good idea what the job entails before buying tools and equipment, it makes sense to have a firm grasp on business requirements before deciding on the details of the technical architecture. Business needs should drive the technology, not the other way around.

And as for those flashy promises… as the poet said, you'll have miles to go before you sleep when you have promises to keep.

Is there a software tool to produce technical specifications from the functional specifications? Isn't there an easier, automated way to come up with the system design?

Some day, you will be able to feed your business requirements into a software tool and have it come up with the Tech Specs. Some day, you will be able to just talk to your computer and have it produce the whole system for you, bug free.

Until that day is here, though, the Project Managers, the Business Analysts and the Technical Leads can remain gainfully employed trying to translate Customer whims into some sort of electronic reality.

Those who lived through the boom – and bust – of computer-aided software engineering tools, understand the promise – and the reality – of such concepts. Until computers learn to think, that job belongs to the people, and the creative process of designing a new system will rest with the humans on the Project Team, regardless of what computer tools they use to produce pretty deliverables.

What is the prototype? Is it a pilot? Proof of concept? System model? Why is it where it is in the lifecycle?

You're absolutely right, the word "prototype" is used to mean a whole variety of things. In the context of the NYS System Development Lifecycle, though, it serves as a low cost (and low risk) technique of validating the technical design and verifying that the business requirements are being correctly translated into system components.

The Functional Specification document has elements of a prototype in it, inasmuch as it contains the printed mock-ups of system interfaces that serve as a first "reality check" of the system design. Then, after the technical architecture and system standards are defined, a prototype of representative system components is developed, using the development tools of choice, if possible. Thus, the "look and feel" of the system is presented to the Customers without a great deal of expense and effort implementing the business rules.

It is where it is in the lifecycle because, generically speaking, that's the right place for it: after the requirements have been finalized and the technical architecture has been defined, but before a major effort has been expended detailing Technical Specifications for every module (and certainly before any real coding effort has occurred!)

How do I establish system standards if there are none? Why do I need them anyway?

Well, there is that old story about the Tower of Babel that sort of makes the point about why a common language ("standards" in the system development parlance) is preferable.

As far as how to go about creating standards when there are none (or worse yet, when the current IT landscape looks like the Tower of Babel), start with "expert opinion": ask competent people, knowledgeable about the chosen technology, what they would recommend (hopefully, some of them are on your team!). Have them write it down, in as much detail as possible, covering at the very least naming conventions, programming standards, and configuration management. Keep refining and augmenting the standards as your team encounters situations not covered by the existing ones.

Keep in mind that the ultimate purpose of the standards is not just having them around but using them, so be strident in your admonitions and vigilant in your QA procedures.

Why plan for data conversion and system deployment in the System Design phase (since neither occurs until much later in the process)?

Ah yes, the old school of "we'll cross that bridge when we come to it." As Project Managers, we would rather go by the "measure twice, cut once" theory. As per the project management

methodology (see Section I, Project Management Lifecycle), planning activities should be complete in the Project Planning phase (which maps to SDLC System Requirements Analysis and System Design phases).

Look at it this way – every system development activity has a chance of affecting other system development activities, and affecting folks outside the Project Team. These effects may range from slight to significant. Now, when would you rather know about them – ahead of time, so you can plan, anticipate, and prepare, or after the fact, so you can scramble, react, and catch up?

What level of detail is appropriate for Technical Specifications?

The theoretical answer is that you should be able to hand off Technical Specifications to a group of developers who have the requisite technical skills but know nothing about this particular system, and receive in return a product that closely matches user requirements.

In other words, the Technical Specifications should stand on their own, providing all of the necessary information to the developer. They should describe the business process in great detail, enumerating ALL pertinent business rules. They should describe the inputs and outputs precisely, providing detailed layouts and specific locations. They should not only mention the database records and elements involved, but also how they should be accessed and updated. They should describe every processing step in great detail, anticipating any question a developer may ask. They should describe every possible user action, and provide detailed instructions on how the system should react to each. They should provide instructions on how to test each module once it's been developed, how to integrate modules into sub-systems, and sub-systems into the system, and how to test the success of each integration.

In the ideal world, upon receiving and reading the Technical Spec, a developer should be able to start – and complete – the assignment without any further questions.

4 SYSTEM CONSTRUCTION

Purpose

System Construction consists of all of the activities required to build and validate the new system to the point at which it can be turned over for System Acceptance. Development efforts in this phase are based on the technical solution created during System Design, which, in turn, was based on the functional and operational requirements captured during System Requirements Analysis.

Included in this phase is the construction of all components of the system, including utilities required to adequately prepare and load the data. In addition, System Construction consists of a series of tests of the system components, with each set of tests being performed against a progressively larger grouping of components until the operation of the system in its entirety has been verified.

Since the ultimate goal of System Construction is to produce a system that is ready for acceptance testing by the Customers, an aspect of this phase is the creation of the various training materials and system documentation that support the new system. These materials need to address both the use and maintenance of the system, and will play an integral part in the System Acceptance and System Implementation phases of the lifecycle.

List of Processes

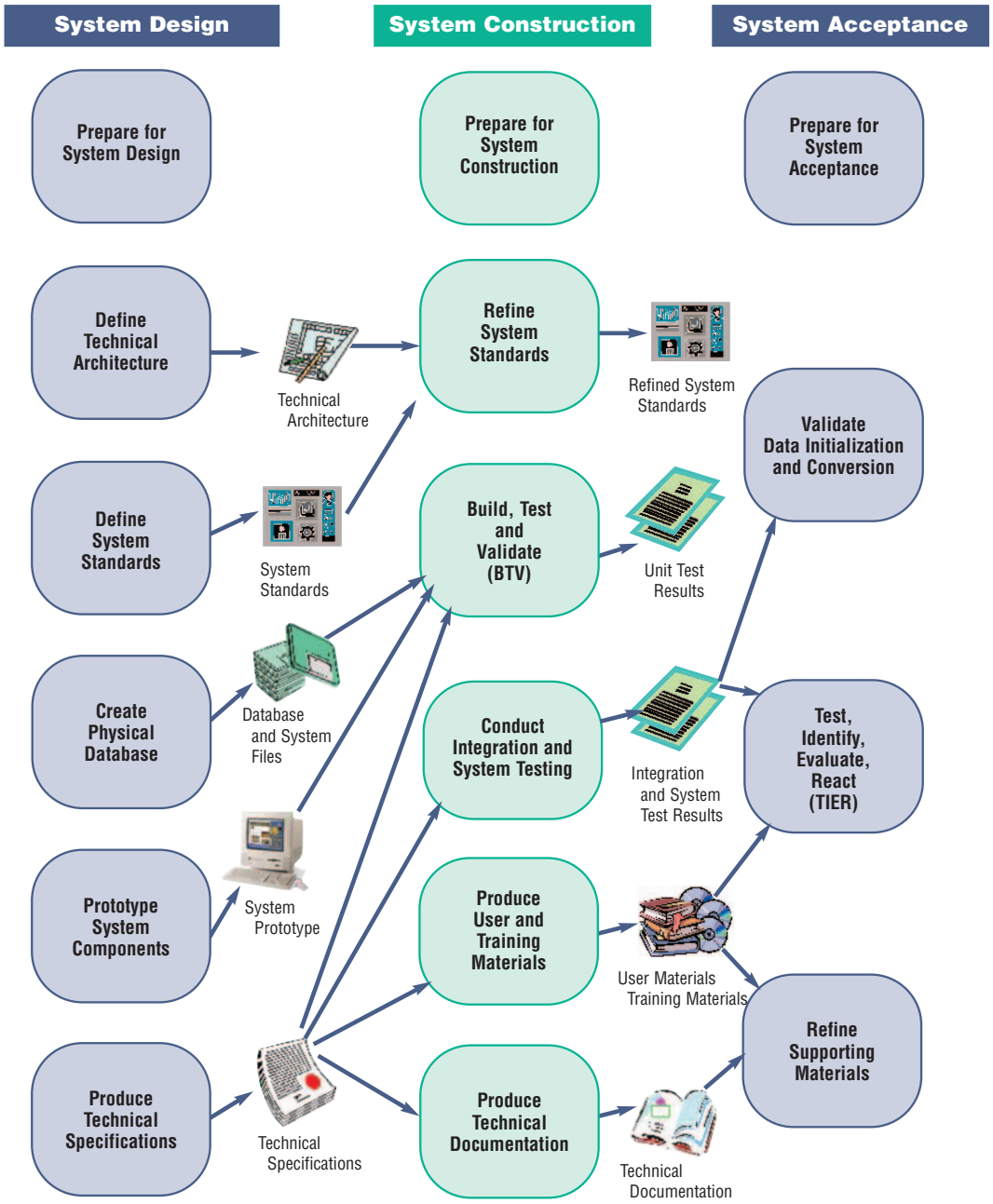
This phase consists of the following processes:

- **Prepare for System Construction,** where the system development and testing environments are established, and where the Project Team is instructed in the processes and tools that will be used throughout this phase.
- **Refine System Standards,** where standards established in System Design are enhanced and adjusted as the team becomes more familiar with the project environment, or in response to changes in the strategic or technical direction of the project.

- **Build, Test, and Validate (BTV),** where individual system components and utilities are constructed and tested to ensure that they perform to the technical and functional specifications.
- **Conduct Integration and System Testing**, where logically related components of the system are assembled and tested as single units, and a complete end-to-end system test is performed.
- **Produce User and Training Materials**, where all Consumer-related documentation and training materials are developed.
- **Produce Technical Documentation**, where all materials intended for the team of individuals ultimately responsible for the on-going maintenance and support of the system are created.

The following chart illustrates all of the processes and deliverables of this phase in the context of the system development lifecycle.

Figure 4-1


System Design
System Construction
System Acceptance
Prepare for System Design
Prepare for System Construction
Prepare for System Acceptance
Define Technical Architecture
Technical Architecture
Refine System Standards
Refined System Standards
Validate Data Initialization and Conversion
Define System Standards
System Standards
Build, Test and Validate (BTV)
Unit Test Results
Create Physical Database
Database and System Files
Conduct Integration and System Testing
Integration and System Test Results
Test, Identify, Evaluate, React (TIER)
Prototype System Components
System Prototype
Produce User and Training Materials
User Materials Training Materials
Refine Supporting Materials
Produce Technical Specifications
Technical Specifications
Produce Technical Documentation
Technical Documentation

List of Roles

The following roles are involved in carrying out the processes of this phase. Detailed descriptions of these roles can be found in the Introductions to Sections I and III.

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Application Developers
- Technical Writer
- Software Quality Assurance (SQA) Lead
- Technical Services (HW/SW, LAN/WAN, TelCom)
- Information Security Officer (ISO)
- Technical Support (Help Desk, Documentation, Trainers)
- Customer Decision-Maker
- Customer Representative

List of Deliverables

The following table lists all System Construction processes, some techniques available for use in executing these processes, and process outcomes and deliverables.

Figure 4-2

Processes	Techniques	Process Deliverables *(Outcomes)*
Prepare for System Construction	Interviews Site Walk-throughs Environmental Assessments	*Established Team and Environment for System Construction*
Refine System Standards	Brainstorming Policy and Standards Reviews Best Practice Assessments Lessons Learned Reviews	Updated System Standards
Build, Test, and Validate (BTV)	Coding Manual Testing Automated Testing Defect Tracking	Unit Test Results *Unit Tested System Components* *Unit Tested System Utilities* *Data Conversion Utilities*
Conduct Integration and System Testing	Manual Testing Automated Testing Defect Tracking Regression Testing	Integration and System Test Results *Validated System* *Validated System Utilities*
Produce User and Training Materials	Technical Writing Illustration On-line Content Development JAD Sessions Prototypes/Content Review	User Materials Training Materials
Produce Technical Documentation	Technical Writing Illustration On-line Content Development JAD Sessions Prototypes/Content Review	Technical Documentation

4.1 PREPARE FOR SYSTEM CONSTRUCTION

Purpose

The purpose of **Prepare for System Construction** is to get the technical environment and the Project Team members ready for successful completion of the full set of System Construction activities.

Description

Much of the preparation for System Construction is completely analogous to that required for System Design, since these phases have many of the same characteristics – potentially expanding team size, introduction of new tools, and the establishment and communication of new processes that must be followed. As with prior phases, it may be necessary to revisit project orientation materials to confirm that pertinent information resulting from the completion of System Design is adequately communicated to individuals joining the team. Additionally, since new development tools and processes may be used in this phase, the training needs of both existing and new team members will need to be assessed.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support

The Project Manager must make sure that the Project Team understands the purpose of new development, management, and testing tools, and the processes that need to be instituted for their use. As the team size grows, so does the potential for mistakes or miscommunications. System Construction often occurs at a point in the Project Schedule when the pressures of meeting deadlines increases. Shortcuts, whether intentional or not, may appear to provide attractive alternatives to meeting commitments. The need to adhere closely to defined procedures is, therefore, even more important than ever before. It should begin in the early stages of System Construction with the education of the team in the processes to be followed.

The start of System Construction marks a point in the project where the overall technical environment becomes more complex and more critical than in prior phases. Due to the scope of activities required to construct and test an application, having access to applicable tools such as automated software development tools, software configuration management tools, testing tools, and defect tracking tools can be extremely valuable to support these efforts.

Due to an increased dependence upon development tools, and the breadth and variety of technical environments that need to be established and supported, sufficient time must be taken at the start of this phase to make sure that these technical environments are correctly installed and configured. This marks the first phase in which it is necessary to institute multiple, distinct technical environments to accommodate the various construction and testing efforts. The environments usually include:

- **Development**, where the individual team members perform their module construction and unit testing activities.
- **Quality Assurance**, where more universally controlled and managed integration and system testing efforts are conducted.

In System Acceptance, an **Acceptance** environment is established which mimics the eventual Production state of the system, and which is able to support load and performance testing. Ultimately, a final **Production** environment will also be needed in which the system will operate once it has been deployed, but this is more traditionally established in System Implementation.

Finally, since many of the System Construction activities involve testing the software being developed, it is important to ensure that everyone, including the Customers, recognize and understand their role in these testing efforts. Preliminary testing efforts should be confined to those individuals involved in the development of the various software modules. Once there is a high degree of confidence in the software, however, further functional testing should include the Customer Representatives to the extent possible. This will provide the greatest opportunity to ensure a correct interpretation of the requirements, and should simplify broader Customer testing efforts downstream in the lifecycle (specifically during System Acceptance). Limiting Customer involvement in these testing activities will introduce risks to the eventual success of the system, and any decision to embrace such an approach should only be made after a thorough evaluation of these risks.

4.2 REFINE SYSTEM STANDARDS

Purpose

The purpose of **Refine System Standards** is to continue to evaluate and evolve the system standards first identified in System Design.

Description

The evolution of system standards, including technical development standards, configuration management standards, and release management standards, continues throughout the entire construction and testing of the application. In most cases, standards refinement occurs as a natural by-product of informal day-to-day interactions between project team members. The Project Manager should encourage this by establishing structured, periodic assessments of existing standards as they apply to the project, and reviews of best practices or lessons learned in the early stages of development that may possibly be applied during later stages.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst

The need to reassess system standards may also be event-driven; for example, in response to a change made in the strategic or technical direction of the project. The decision to purchase components of the system instead of developing them from scratch, and the assessment of the impact of this decision, may warrant a re-evaluation of existing standards. Similarly, a proof-of-concept prototyping effort may yield results that indicate the need for a technical solution that is different from what was originally envisioned. Examining standards at such a time will enable the team to reconfirm their applicability to the project given the new circumstances.

While many Project Teams view this process as primarily focused on the refinement of technical standards that apply directly to the system being developed, the same evaluation and appraisal process can be applied to any of the processes or practices being followed by the development team. Periodic reviews and refinement of all such processes can help reduce project risks and ensure that the processes culminate in the desired results. Depending upon factors such as team size or project complexity, reviews can range from half-day facilitated

sessions to a simple five-minute discussion once a month during regular team meetings. The Project Manager must determine the approach that is most appropriate for each project. It is more important that the reviews are done, than *how* they are done.

Deliverable

- **Updated System Standards** – An updated document detailing the various standards and conventions to be applied and adhered to throughout the project lifecycle.

4.3 BUILD, TEST, AND VALIDATE (BTV)

Purpose

The purpose of the **Build, Test, and Validate (BTV)** process is to produce a complete set of software modules, and to perform the first round of validations on these components.

Description

As the name also implies, this process can be decomposed into three tightly coupled, and often parallel and iterative, subprocesses, as follows:

Build

The physical construction of the system components and utilities takes place during the Build portion of this process. In order to manage this effort, the Project Manager must have an exhaustive list of the modules to be built. With Tech Specs defining this list, development work can be logically partitioned (both in terms of identifying related work packets when distributing the work across the Project Team, and in terms of determining the sequence in which the development efforts will be approached), and progress can be measured and reported.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Customer Representative

Partitioning system components and deciding the order in which their development will take place is something that the Project Manager and Technical Lead should pay particular attention to. Done correctly, this can simplify the tracking and reporting of the team's progress to your Customer. In addition, it can enable the team to focus on more difficult or challenging areas of the system first, or on those areas upon which subsequent development efforts may be dependent. One last benefit is that by defining workable components of the system that can be built and demonstrated as a unit, it may be easier to involve the Customer in reviews, walk-throughs, and checkpoints. This is a great means of gaining early buy-in and enthusiasm for the system, and also for confirming that both the Project Team and Customer share a common vision and understanding of the system.

In order for the Application Developers to be able to code each module, they must have access to the Technical Specification associated with that module. Since it is likely that some of the Developers may not have been involved in creating these specs, the Business Analyst(s) should be available to answer questions dealing with the desired functionality and the Customer's intent behind the specs. Similarly, the Technical Lead should provide the technical background and expertise that the Application Developers may lack.

Test

With unit test plans created during System Design, the individual who developed the code typically performs unit testing of each module. Establishing a process in which this unit testing is performed independent of the developer may improve the quality of the test, but may also be impractical given staffing or schedule limitations. It is important that this testing be performed thoroughly, to validate each of the logic paths the software module needs to support, and to capture the results of the tests for future reference. Unit testing is usually performed within the development environment; however, specific actions may need to be taken to ensure that this environment is initialized with the appropriate data and test tools. The test plans should identify any conditions required prior to the start of the unit testing efforts.

In communicating the standards and specifics to which each module must conform, along with the criteria by which it will be tested, it may be useful to develop checklists for each programmer to use during development, and for each team member to use when performing QA of other team members' code. Since common routines may have different requirements or standards than GUI modules, which in turn may differ from other types of components, it may be appropriate to develop checklists specific to the type of module being developed.

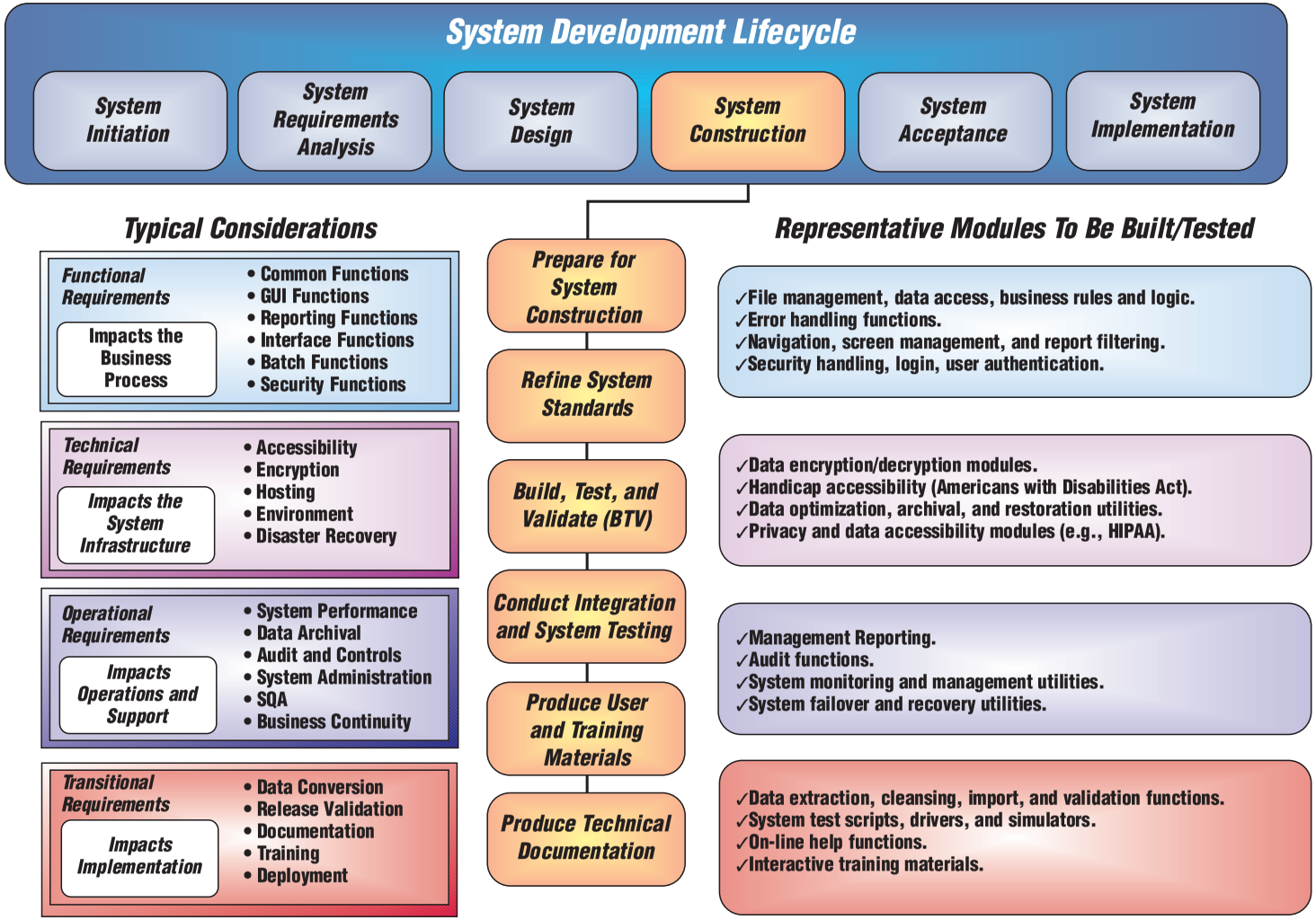
Validate

Validation consists of comparing the actual results of the testing against the expected results that were identified before any testing was performed. By putting these two sets of results side by side, the developer can determine if any corrections are required in the software. If so, another iteration of the build, test, and validate activities begins. This is also a point in the project when the Project Manager and Technical Lead must employ the concept of "peer reviews", in which team members review each other's code to confirm that the appropriate conventions and standards are being followed.

Peer Review is an indispensable tool for the BTV process. It reinforces the deliverable schedule, it promotes dissemination of technical and functional expertise, it advances the rapid implementation of best practices and understanding of lessons learned, and it is an invaluable early warning system for the Project Manager, alerting him or her to problems with the people, the process, the system, and the technology. From the perspective of team chemistry, it is important that the emphasis of these peer reviews be on providing constructive criticism that will result in the eventual building of a more robust and technically sound solution.

Understanding the overall premise of BTV, it is important to realize that this process must be applied for all types of modules being constructed.

Figure 4-3 System Construction Considerations



As more and more of the system is developed, it will be valuable to involve the Customer as much as possible so that adjustments can be accommodated early in the development cycle. Again, the functional aspects of the system are those that the Customers and Consumers can most closely associate with and relate to their day-to-day responsibilities. Frequent interaction with the system at this point in the process will enable them to visualize what it is they will be getting, and will help them to begin to adjust to and accept the changes that they will be faced with once the system is live.

While it is beneficial to get as much feedback as possible, it is also critical that the Project Manager differentiate between changes required to satisfy the functional requirements agreed to during System Requirements Analysis and changes that would result in alterations to the scope or direction of the project.

The most common cause of system development problems is "scope creep," arising from either a continuous adjustment of desired functionality by the Customer, or from an incorrect or incomplete analysis and design effort. The proper course of action is to invoke the change control process in the Project Management Lifecycle, and address the issue at its source, wherever it may be.

What you want to avoid at all cost is "team thrashing", where the developers are continuously buffeted with "new, improved" specs based on the latest understanding of user requirements.

Deliverable

- **Unit Test Results** – A comprehensive set of output identifying the individual unit tests that were performed, along with the detailed outcomes of each test. These test results are contained in the Unit Test Plan section of the Technical Specifications.

4.4 CONDUCT INTEGRATION AND SYSTEM TESTING

Purpose

The purpose of **Conduct Integration and System Testing** is to continually validate larger and larger combinations of modules until the entire system is operating correctly as a single unit.

Description

Having validated individual software components in Build, Test, and Validate, it is now time to begin validating the interaction among these components. The sequence and manner in which these software modules are combined should be defined in detail in the Integration Test Plans that were built during System Design. Where unit testing focused on the individual elements (as decomposed in the Technical Specifications), integration testing now needs to "roll up" the elements into logical groupings (Integration Test Packets) and sub-systems (as identified in the Functional Specification).

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- SQA Analyst
- Technical Services
- Customer Representative

Much of System Construction involves an iterative set of processes where the results of one activity may seed efforts in related activities. Results of integration testing may identify components of the system with defects that require re-execution of the Build/Test/Validate process; and once software modules have been modified and retested at the unit level, regression testing of the modules and any subsystems to which they pertain will again be required.

This is often a good point in the project to begin to capture metrics relating to the quality of the system being developed. Data relating to the number of defects identified in testing, the types of defects, criticality, etc., can all be useful in understanding the state of the new system and attempting to evaluate deficiencies in the development processes being followed. For example, if the same errors are repeatedly detected from release to release, it may be an indication that aspects of the quality assurance processes being applied are not adequate, or that the unit tests for a particular module are not stringent enough to fully validate its functionality. (Refer to Figure 4-4 for a sample Defect Log.)

It is essential to know exactly which versions of each software component are being tested in each release. Software Configuration Management tools can be used to manage these releases, allowing the Project Team to know the exact contents of the release being tested. By tightly controlling the testing environment, it is easier to identify the specific causes of any suspected defects in the system.

It may be useful to perform "root cause analysis" of the problems identified during integration testing. While all of them by definition are "system problems," some of them are straight-forward technical or system bugs, while others may indicate issues with the technical architecture, inaccuracies in the original requirements gathering, or deficiencies of application design, each of which requires a different course of action.

Ultimately, completion of this process should result in one of two conditions:

- All Integration Tests have been successfully executed, demonstrating that the system performs to the expectations of the Project Team, or
- The majority of Integration Tests have been successfully executed, identifying a known set of defects, which will require further resolution at some point in the future.

Should the latter situation arise, there may come a point at which it is appropriate for the Project Manager to discuss with the Customer the value of continuing this process at the expense of delaying the start of System Testing. If the known defects are relatively insignificant and do not prevent the system from satisfying the overall business objectives for which it was developed, it may be completely acceptable to initiate System Testing knowing that there may be some remaining Integration Testing issues to be resolved.

Having completed Integration Testing, System Testing now serves two purposes. First, it is the ultimate Integration Test, incorporating all modules into a single system. Second, it validates the operation of the system as it performs against the anticipated boundary, volume, and peak load conditions. Completion of System Testing results in a similar decision point to that which was encountered at the end of Integration Testing.

If any known defects exist, the Customer must be consulted to determine whether they are of sufficient significance to prevent the team from progressing into System Acceptance. If the known problems are sufficiently minor, then it may be reasonable to advance into System Acceptance knowing that there may be some remaining System Construction activities.

Deliverable

- **Integration and System Test Results** – A comprehensive set of completed test plans identifying all integration and system tests that were performed, along with the detailed outcomes of these tests, the list of defects identified as a result of these tests, and the results of any subsequent retests. These test results are contained in the Integration and System Test Plan sections of the Technical Specifications.

Figure 4-4 Defect Log

< Name of Agency >

Defect Log for <Testing Performed>

< System Name >

Agency	
Project Name	
Project Sponsor	
Project Manager	
Document Date	
Prepared By	

*Enter the name of the **Agency** for which the system is being developed.*
*Enter the **Project Name**, and the names of the **Project Manager** and the **Project Sponsor**.*
*Enter the **Date** as of which this document is current.*
*Enter the names of the Project Team members by whom the document was **Prepared**.*

Figure 4-4 (continued)

Defect Log for <Testing Performed>

Testing Performed: _____

Defect Identification			Test Results:			Defect Resolution:			Approval		
Defect #	Status	Test Case #	Tester	Date	Defect Description	Application Developer	Resolved Date	Resolution Description	Re-Tester	Date	Approved Date

*This form is used to log defects encountered when Integration, System, Data Validation or Acceptance **Testing** is **Performed**.*
Most of the information on this form comes from test results recorded on Integration, System, Data Validation, and Acceptance Test Plan templates (see Technical Specifications document template).
***Defect #** is assigned sequentially as defects are discovered.*
***Status** is either Open or Closed (following the Approval Date),*
***Test Case #, Tester** and **Date** refers back to the Test Plan.*
***Defect Description** is a narrative of how the test results differ from the expected results.*
***Defect Resolution** and **Approval** data comes from the applicable Test Plan.*

4.5 PRODUCE USER AND TRAINING MATERIALS

Purpose

The purpose of **Produce User and Training Materials** is to create materials to train the Consumers in the use of the system which can be used as reference materials once the system has been implemented.

Description

This process often overlaps with Build, Test and Validate, and even more commonly with Conduct Integration and System Testing. Materials may vary in format and level of detail, depending upon their anticipated use, but the important thing is that they be developed according to the expectations of the Customer. Specifically, System Requirements Analysis and System Design should have resulted in a detailed definition of the level of documentation and training materials needed.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Technical Writer
- Customer Decision-Maker
- Customer Representative

This is frequently the point in the lifecycle at which a Technical Writer may be introduced onto the Project Team. It is important to time the addition of the Technical Writer to the point in the Project Schedule at which the definition of the system is solid and unlikely to change significantly. As an example, the development of the on-line help that supports a set of screens and reports should only occur after consensus has been reached with the Customer regarding their content, layout and design. Developing help files prematurely could result in rework.

As the project progresses into System Acceptance, the documentation materials will be provided to the individuals responsible for performing the testing and validation of the system on behalf of the Consumers. Prior to System Acceptance, it is the responsibility of the Project Team to assess the quality and content of these materials, and to ensure that changes to the system resulting from Integration and System Testing efforts are accurately reflected in both the reference and training materials.

Regardless of whether or not the development of training materials requires actual software development (as would be the case should the system include automated or computer-aided

training modules), the production of user and training materials still follows the same Build, Test, and Validate cycles as the rest of the system. As each component is created, it should be evaluated against predefined criteria to ensure that it delivers the instructions and concepts it is intended to deliver. Similarly, appropriate quality assurance reviews must be performed to validate that correct standards and conventions are being followed.

Ideally, Customers will have been involved throughout the Integration and System Testing process. When finalizing the content of the User and Training Materials, it is often useful to take advantage of their experiences with the system by involving these same Customers in the evaluation of the structure and content of these materials. Having already experienced the system first-hand, their insights can prove invaluable in assessing the usefulness of these materials.

Deliverables

- **User Materials** – A collection of documents, either hard-copy or on-line, designed to assist Consumers in the use and operation of the application.
- **Training Materials** – Materials, for Consumers who are less familiar with the system, that provide instructions on the intent and use of the system.

4.6 PRODUCE TECHNICAL DOCUMENTATION

Purpose

Produce Technical Documentation consists of assembling and organizing all technical information and materials that will be needed for the long-term support, maintenance, and operation of the application.

Description

As with training materials and Consumer-oriented documentation, the level of detail and the format of technical documentation will vary from project to project. The audience for this

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Technical Writer
- SQA Analyst
- Technical Services
- Information Security Officer
- Technical Support

documentation often includes Help Desk personnel and Technical Services, as well as people who will ultimately maintain this system. One approach to developing technical documentation is to revisit prior Technical Specifications and supporting materials (in the event that the development tools used do not automatically update this information). Depending upon the rigor and approach taken during System Design and Construction, a review of these specifications may determine that updates are needed to accurately reflect design changes introduced as the system was being built and tested. If this is the case, the Project Manager may need direction from the Project Sponsor as to whether or not the benefits derived from updating these materials warrants the investment.

Minimally, the logic and execution of the overall application should be documented. How this is accomplished should be based on organizational standards, either by updating existing documentation or by creating new and equivalent documentation. When in doubt regarding the correct approach, the SQA Analyst should be consulted for an understanding of the policies and conventions that should be followed.

Deliverable

- **Technical Documentation** – A collection of materials defining the technical aspects of the system. The intent of this documentation is to provide the team of individuals ultimately responsible for the application with a level of understanding sufficient to enable them to successfully operate and maintain it.

Measurements of Success

The success of the System Construction phase can be most expeditiously measured by the project variance – how closely the actual construction of the system follows the plan and the schedule. The ultimate measurement of success, of course, is the acceptance of the system by the Customer.

During each process, the Project Manager can assess how successfully the project is proceeding by utilizing the measurement criteria outlined below. More than one "No" answer indicates a serious risk to the eventual success of the project.

Figure 4-5

Process	Measurements of Success	Yes	No
Prepare for System Construction	Do your team members agree that they are positioned with the appropriate skills and training to perform the System Construction activities, or that a plan for addressing any gaps has been identified and communicated to them?		
	Do your team members agree that they have access to the appropriate systems and repositories?		
Refine System Standards	Has the Technical Lead conducted a review of the software configuration management process with all the team members?		
	Has the Technical Lead conducted a review of the Build/Test/Validate process (including peer reviews) with all the team members?		
Build, Test, and Validate (BTV)	Is the Project Team encountering no delays due to the environment not being ready or operating incorrectly?		
	Are new team members becoming productive within a week of joining the team?		
	Has the rate of defects declined at least 50% by the second BTV cycle?		
Conduct Integration and System Testing	Are the incremental application "builds" occurring smoothly?		
	Is the integration testing process resulting in low levels of rework (either due to technical deficiencies or due to design changes)?		
	Have the representatives of each Customer functional group had more than one opportunity to review the system deliverables to date?		
Produce User and Training Materials	Has the Customer agreed to the system training approach?		
	Has the Customer signed off the on the User and Training materials?		
Produce Technical Documentation	Do the team members producing technical documentation have an example to follow that is considered a model for technical documentation throughout the organization?		

Phase Risks / Ways to Avoid Pitfalls

PITFALL #1 – THROWING THE QA OUT WITH THE SCHEDULE

Most frequently, system development efforts encounter their first major delays in System Construction. This is where the proverbial rubber hits the road, exposing lack of technical expertise, weakness of design, laxity in requirements definition, and (why not go all the way back!) even errors in judgement during project selection and initiation.

The knee-jerk reaction of a green (or insecure) Project Manager is to prune the schedule of all "extraneous" tasks and whip the developers into a 24-hours-a-day frenzy of code production. Inevitably, the first casualties are the quality assurance activities. They "steal" precious time needed to re-write the ill-fitting module for the fifth time, and they do nothing to arrest the inexorably receding date of the latest elapsed milestone.

The reality, of course, is that in by-passing QA, the Project Manager is making sure that the system will not only be delivered behind schedule (if ever!) but also will not work correctly. Instead, the better course of action is to find and address the root cause of the apparent problem, recast the schedule using realistic assumptions, and reset expectations all around.

PITFALL #2 – HIDING IN THE "BLACK BOX"

System Construction invites the least amount of involvement from the Customer and Consumer community. After all, there is no "finished" product to show off, and the work itself is the exclusive province of the adepts of the complex arts of information technology, with their own acronymic language, arcane conventions, and peculiar interactions. The temptation is to let the team turn to itself, except for a necessary occasional interface with the data keepers or the server masters.

Not a good idea! No matter how perfect the prototype, how meticulous the Technical Specs, or how complete the requirements, during the course of construction inevitably something will come out differently, some questions will arise, and some issues will be identified. The longer the issues fester, the questions simmer, and the changes multiply, the greater will be the dichotomy between what was ordained, and what was produced.

PITFALL #3 – USING THE "PARTS IS PARTS!" APPROACH

Coming out of System Design, the Project Manager is faced with scores and scores of Technical Specs. Some are longer, and some are shorter, but they all seem pretty interchangeable, and frequently a Project Manager will make staff assignments based on some seemingly deliberate, but really arbitrary order – functional, technical, or using the system design document table of contents.

However, to ensure smooth integration testing and to maximize the utility of the peer and Customer reviews, it behooves the Project Manager (with the help of the Technical Lead, of course) to partition the development into discreet work packets that can be built, tested, and integrated independently, by small teams (or even single individuals).

Equally important is the proper sequencing of such work packets. High-risk, critical packets should always be developed first, to quickly head off major problems, avoid false confidence, and prevent unpleasant surprises.

PITFALL #4 – WHO NEEDS REAL DATA?

Priming the database with real base tables, and loading real data is a laborious, messy process, which seems especially more so for a mere test environment. With the proliferation of test data-generating tools, the temptation is to throw some plausible as well as some extreme values into the database, and let it rip.

Unfortunately, there are two problems with using this method exclusively. First, seeing no recognizable data on screens and in reports, the Customer is unable to relate to it in any meaningful fashion, and thus can only add limited value to the review process. Made-up data shifts focus from the content to the format, and makes fonts and colors have greater importance than algorithms and business rules.

Second, (paraphrasing Ken Orr,) data reflects reality, and reality knows no standards. No matter how accurate your prior analysis of data domains and the like, the real database that you will have to convert and load before the system becomes operational will bestow you with certain surprises.

Data specially generated to test certain conditions definitely has its place in both unit and system testing scenarios; but nothing can replace live, messy and impetuous, but inescapable and familiar, real data.

PITFALL #5 – THE 90% COMPLETE SYNDROME

Remember Zeno's paradox, wherein a runner in a 100-yard dash can never arrive at the finish line, because first he must run _ of the race, which leaves him 50 yards from the finish line; then he must run _ of the remaining race, which still leaves him 25 yards away; and as he continues in this fashion ad infinitum, he can never cross the finish line because he is always _ of the remaining distance away from it! It seems like many tasks enter the same sort of the twilight zone: they get to the 90% complete mark, but never quite make it to the "done with" category.

You have two weapons to combat the 90% complete syndrome. First, instead of asking what percentage is complete, ask how much effort is LEFT to be totally "done with" the task; that focuses the mind away from the hopeful, optimistic and eager-to-please 90%-complete assumption, and on the actual, realistic estimate of the remaining work. It also sets up an unambiguous test – if the person estimates 8 hours left, spends 10 hours on the task, and still tells you he needs another 8 hours to finish it, it's time to take out the second gun: third-party verification. Have another technician check out the work done, verify estimate to complete (ETC), and then take whatever action is appropriate.

PITFALL #6 – THE BIGGER THEY ARE, THE HARDER THEY FALL

Sometimes the size of the system, rather than its functional or technical complexity, becomes the primary challenge for the Project Manager.

Large systems are understandably and reasonably broken into many sub-systems, each comprising a number of modules. The Project Team is likewise partitioned into many smaller groups, some with only a single individual assigned to a single unit of work.

The challenge comes in coordinating all those small groups, and fitting the results of all those small units of work, together. A Project Team may resemble a connectionist network, where every node is connected bi-directionally to every other node; as the number of nodes grows, the number of connections grows much more rapidly: while only 2 connections exist between 2 nodes, 4 nodes require 12 connections, and as few as 12 nodes have 132 communications channels going all at once!

The same is true for integrating the work products. The groups may lose the "big picture" view of the project, and concentrate on their myopic, module-level view of the effort, which is likely to deviate from the application baseline. In the end, even if the products are physically compatible, they may need to be reworked to conform to a single standard.

A seasoned Project Manager will set up clear hierarchical channels of communication (limiting the number of nodes without limiting the flow of information!), encourage peer-to-peer communications at the Technical Lead level (leveraging collective expertise), and require peer reviews across development groups (reinforcing standards, disseminating best practices, and sharing lessons learned).

Frequently Asked Questions

Training and documentation are not IT's job. Why are these processes and deliverables part of the system development lifecycle?

This ***Guidebook*** is a firm proponent of the principle "If you want a professional job, hire a professional" and does not advocate forcing programmers to become trainers (or writers.) However, the responsibility for getting the Consumers trained, and for providing clear and helpful documentation to the Performing Organization, still rests with the project – and the Project Manager.

The Project Manager must add to the Project Team requisite resources to prepare and provide training, and to create technical and user documentation, whether those resources come from a separate technical writing group, are contracted from a training vendor, or developed by tapping into the hidden talents of the chip-heads.

What's a test script?

A test script is a series of instructions that guides someone involved in unit, integration, system or acceptance testing activities. Following a test script, the tester sets up a series of scenarios wherein certain conditions and inputs produce a specified expected result. Whether the results differ from stated expectations or confirm them, the test script provides instructions for recording and reporting variance as well as conformance.

Test scripts need to be comprehensive but also practical. They should test all possible types of state/input/process/output combinations, but not necessarily all values within each type (for example, if you are testing for the module's ability to detect odd numbers, it is not necessary to submit EVERY odd number to the test; a small representative sample of odd and even numbers should suffice).

Where in the lifecycle do you do a stress test?

Stress testing is usually done at the sub-system level, after system components have been integrated, and repeated again at the system level. Stress testing checks the system's ability to handle abnormally large (but theoretically possible) volumes of transactions, and to recover from abnormal conditions (such as power outages).

5 SYSTEM ACCEPTANCE

Purpose

System Acceptance is the point in the lifecycle at which every aspect of the application being developed, along with any supporting data conversion routines and system utilities, are thoroughly validated by the Customer Representatives prior to proceeding with System Implementation.

This entire phase is centered around gaining sufficient evidence of the system's accuracy and functionality to be able to proceed to System Implementation with the highest level of confidence possible in the success of the system. This phase differs from System Construction in that acceptance testing is the final opportunity to establish that the system performs as expected in an environment that mimics production as closely as possible. In addition, while the Customer Representatives were certainly engaged throughout prior testing efforts, they now assume an even more critical role in the testing efforts in that they now need to exercise the system in the same way that they will once the full system is implemented. With the testing roadmap established in earlier lifecycle phases, the Customer Representatives now take responsibility for maneuvering the system through its operations.

In addition to confirming the operation of the system and its fit to the business needs that it is intended to satisfy, System Acceptance is also the point in the lifecycle during which all supporting documentation and reference materials are updated to guarantee their consistency with the final delivered system.

List of Processes

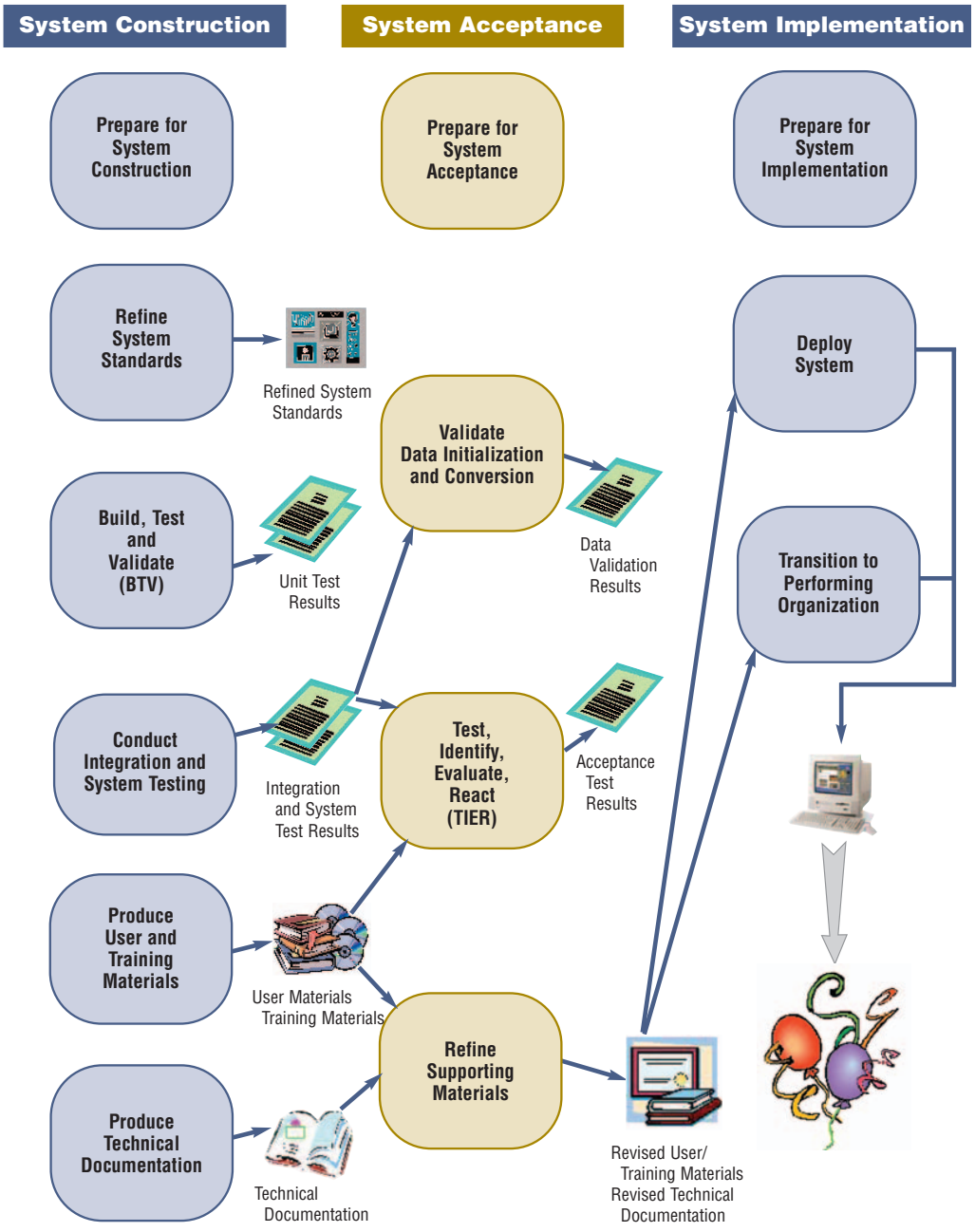
This phase consists of the following processes:

- **Prepare for System Acceptance,** where the system acceptance environment is established, and where the testing team is instructed in the use of processes and tools necessary throughout this phase.
- **Validate Data Initialization and Conversion,** where the processes and utilities used to populate the system database are tested to confirm that they provide a starting point from which the new system can begin processing.

- **Test, Identify, Evaluate, React (TIER),** where the system functions and processes undergo a series of exhaustive acceptance tests to validate their performance to specifications, and where examination of test results determines whether the system is ready to begin production.
- **Refine Supporting Materials,** where the various materials that support the use, operation, and maintenance of the system are updated to reflect any necessary adjustments resulting from acceptance test results.

The following chart illustrates all of the processes and deliverables of this phase in the context of the system development lifecycle.

Figure 5-1


System Construction
System Acceptance
System Implementation
Prepare for System Construction
Prepare for System Acceptance
Prepare for System Implementation
Refine System Standards
Refined System Standards
Validate Data Initialization and Conversion
Deploy System
Build, Test and Validate (BTV)
Unit Test Results
Data Validation Results
Transition to Performing Organization
Conduct Integration and System Testing
Integration and System Test Results
Test, Identify, Evaluate, React (TIER)
Acceptance Test Results
Produce User and Training Materials
User Materials Training Materials
Refine Supporting Materials
Produce Technical Documentation
Technical Documentation
Revised User/ Training Materials
Revised Technical Documentation

List of Roles

The following roles are involved in carrying out the processes of this phase. Detailed descriptions of these roles can be found in the Introductions to Sections I and III.

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Application Developers
- Technical Writer
- Software Quality Assurance (SQA) Lead
- Technical Services (HW/SW, LAN/WAN, TelCom)
- Information Security Officer (ISO)
- Technical Support (Help Desk, Documentation, Trainers)
- Customer Decision-Maker
- Customer Representative
- Stakeholders

List of Deliverables

The following table lists all System Acceptance processes, some techniques available for use in executing these processes, and process outcomes and deliverables.

Figure 5-2

Processes	Techniques	Process Deliverables *(Outcomes)*
Prepare for System Acceptance	Interviews Site Walk-throughs Environmental Assessments Acceptance Test Plan Review	*Established Team and Environment for System Acceptance*
Validate Data Initialization and Conversion	Manual Testing Automated Testing Defect Tracking Regression Testing	Data Validation Test Results *Validated Data Initialization and Conversion Software*
Test, Identify, Evaluate, React (TIER)	Manual Testing Automated Testing Defect Tracking Regression Testing	Acceptance Test Results *Validated System* *Validated System Utilities*
Refine Supporting Materials	Technical Writing Illustration On-line Content Development Content Review	Revised User/Training Materials Revised Technical Documentation

5.1 PREPARE FOR SYSTEM ACCEPTANCE

Purpose

The purpose of **Prepare for System Acceptance** is to ensure that the testing environment to be used during System Acceptance is ready and operational, and to take any steps needed to prepare the acceptance testing team to successfully achieve their testing goals.

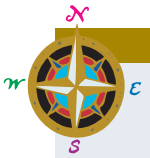
Description

This phase of the SDLC is significant because it is the last time that rigorous testing will be performed on the system before it goes into production. It is also very likely the first time that Customer Representatives will be able to exercise the application in a near-production environment, which adds a unique perspective to the testing efforts.

Preparation of both the testers and the environment in which they will operate is crucial to the success of this phase. User and training materials must be distributed in advance of this effort, and any training sessions needed to familiarize the testers with the application must be conducted.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Application Developer
- SQA Lead
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders



In an ideal situation, those participating in the testing should receive the exact training and materials intended for Consumers, so that the usefulness and acceptability of the materials can be validated.

In addition to familiarizing the testing team with the system, preparatory efforts must clarify for the team all testing roles and responsibilities, the timeline allocated to these efforts, and all processes to be followed regarding recording of testing results and reporting of defects. Although prior testing activities should have included Customer Representatives as part of the test team, it is common for this team to include an increased number of representatives so that real production operations can be better emulated. As a result, the testing team may now consist of participants who may not be as accustomed to rigorous testing activities as were members of the integration and system testing team, who typically have a more systems-oriented background. Therefore, expectations of these individuals need to be clearly defined, as do such elements as the testing strategy to be followed, the extent of testing to be performed, the definition of acceptance, etc.

Preparation of the environment in which acceptance testing will be performed is primarily focused on confirming that it is as close to the production environment as possible and on migrating the application from the QA to the Acceptance environment.

5.2 VALIDATE DATA INITIALIZATION AND CONVERSION

Purpose

The purpose of the **Validate Data Initialization and Conversion** process is to confirm before the system begins production that all utilities and processes needed to load data into the system work correctly, and that any data carried forward from another system is migrated completely and accurately.

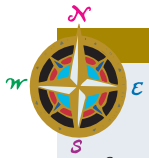
Description

As important as it is to ensure that the new application functions properly, it is equally important to ensure the accuracy of the data being processed by the system. This effort starts with the initial loading of data, also known as “Day 1” data. The data is most often populated using two main methods – the manual loading of information required by the new system that cannot be extracted or obtained from an existing system, and the automated loading of information currently available in one or more existing data sources.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Application Developer
- SQA Lead
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

The acceptance testing team must exercise all aspects of the data initialization and loading of information into the system database. Testing of the data load should be conducted very much like the testing of the application itself, with all participants capturing the test results and identifying any defects. The goal is to determine whether the quality of the data load process and the resulting data are sufficient to proceed with implementing the system. Any data problems that jeopardize the eventual success of the system clearly need to be addressed. It may be perfectly acceptable, however, to advance into further application testing activities with a known set of low-impact data problems, as long as the impact of these defects on subsequent testing efforts is understood in advance, along with a defined timeframe by which the errors need to be corrected.



The key difference between acceptance testing activities and all prior testing efforts is that while it was reasonable to expect iterative testing cycles in earlier phases, the goal of acceptance is to demonstrate that the deployment and use of the system will be successful in a production-like setting. Therefore, whether validating data initialization efforts or specific system functions (as described in the following process), all activities performed in this phase should already have been successfully demonstrated in System Construction, albeit in a slightly different environment.

This does not mean that that there won't be decision points throughout this phase at which test results will need to be evaluated, usually as part of an overall set of test results, to determine the proper course of action. Once these test results are in hand, then an informed decision can be made to either move ahead with continued testing, or to address known issues as they are discovered, only moving forward when the error condition has been corrected.

Deliverable

- **Data Validation Test Results** – A comprehensive set of completed test plans identifying all data initialization and conversion tests that were performed, along with the detailed outcomes of these tests, the list of defects identified as a result of these tests, and the results of any subsequent retests. These test results are contained in the Acceptance Test Plan section of the Technical Specifications.

5.3 TEST, IDENTIFY, EVALUATE, REACT (TIER)

Purpose

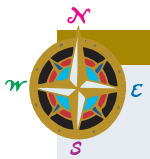
The purpose of the **Test, Identify, Evaluate, and React** process is to execute a complete suite of tests against the application in a production-like environment, assess the results of the tests, and determine whether it is appropriate to proceed with System Implementation, or whether corrective actions are required to address any defects encountered.

Description

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Application Developer
- SQA Lead
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

This process is analogous in many ways to the Conduct Integration and System Testing process in System Construction. While the primary responsibility for conducting the testing has moved from the Application Developers to the Customer Representatives, many of the principles that applied to earlier testing efforts apply here as well. The need for capturing testing metrics remains essential for conducting quality assurance practices, and the adherence to rigorous configuration management and release migration procedures remains crucial to understanding exactly which versions of the software are being tested at any given time.



Because the Customer is anxious to implement the new system and restore testing personnel to their primary business functions, acceptance testing tasks are often underemphasized. Thorough testing procedures cannot be stressed strongly enough. Failure to perform these tasks with high quality and attention to detail could cause serious problems in the future, perhaps after the system has been placed into production. Time invested at this stage will save time overall.

Throughout this process, any problems identified by the testers must be recorded and tracked to closure on defect logs. Continual interaction is essential between those doing the testing and those who developed the application. The Project Manager must closely manage the activities in order to ensure adherence to the Project Scope and Schedule.

Another factor to consider during this process is that organizations may choose to perform parallel operations during acceptance testing. This requires a continuation of existing business processes at the same time that the new system is being tested. This may mean that while transactions are entered into the new system, they will also need to be entered separately into any existing legacy systems, or may need to be captured in whatever manual systems are currently being utilized. This often requires additional staff or extra hours in order to keep up with the additional workload, but allows the results of the two processes to be compared for accuracy. If this parallel approach is taken, the extra burden on the Performing Organization will need to be estimated and communicated to the Stakeholders so that they can make an informed decision regarding any additional costs, the duration for which the organization can sustain these costs, and the benefits resulting from this approach.

Regardless of whether or not parallel operations are planned, the recommended approach for testing applications is to drive a time-boxed set of coordinated tests that adhere to the TIER approach, as follows:

Test: Following the initialization of the Acceptance environment, acceptance testing will occur, during which all perceived defects in the application are recorded. The exact frequency with which these defects are reported will vary with each project – the important point to note here is that communication of these defects needs to be constant throughout the testing to avoid the 'big bang' effect that can occur when all issues are reported only upon completion of the testing.

Identify: The Project Manager will engage the appropriate team members to analyze each reported defect to determine the cause of the defect being reported, and to identify whether or not a true system error has been encountered. While defects are often related to adjustments needed in the application software, it is equally possible that the root cause of a reported

defect is the tester's misunderstanding of exactly how the system was designed to operate. Changes to normal business operations due to new functionality, combined with the revised look and feel of the application, often result in system errors being reported that in fact are examples of the system working exactly as designed.

The Project Manager should keep in mind that system errors or defects may be reported that result more from a misinterpretation of expected functionality as opposed to a technical defect. Even though the system may be operating exactly as defined, this scenario may point to other non-technical errors associated with the system. It may be possible that the on-line help system may not sufficiently describe the system's operations, or that a component of the overall training package may require an increased level of detail in one or more areas. Take advantage of System Acceptance to evaluate the entire system and its supporting materials, and make adjustments now while you can still get out in front of the final implementation.

Evaluate: If a defect in the application is identified, the Project Team will work together to identify the appropriate corrective action. A decision will be made regarding whether or not system modifications are required, whether data loaded in the prior process needs to be corrected, whether operational procedures need to be adjusted, or whether some other form of correction is required. Once a corrective action is identified, it will then be prioritized, along with all other on-going activities, to determine if this issue is of sufficient impact to warrant adjustments being made during System Acceptance. Since all testing efforts should adhere to the Project Schedule, the underlying question becomes, "Can the system be placed into production with the existence of this condition, or is its impact such that implementation of the system is not possible due to inability to perform essential business operations"? If an acceptable work-around exists, or if the impact is minimal, then a determination can be made to handle the correction as part of a normal production support issue once the system is implemented.

React: Once the appropriate actions and priorities have been identified, the defect will be resolved. For those defects requiring changes to the application, the appropriate changes should be made, tested, and re-released to the Customer Representa-

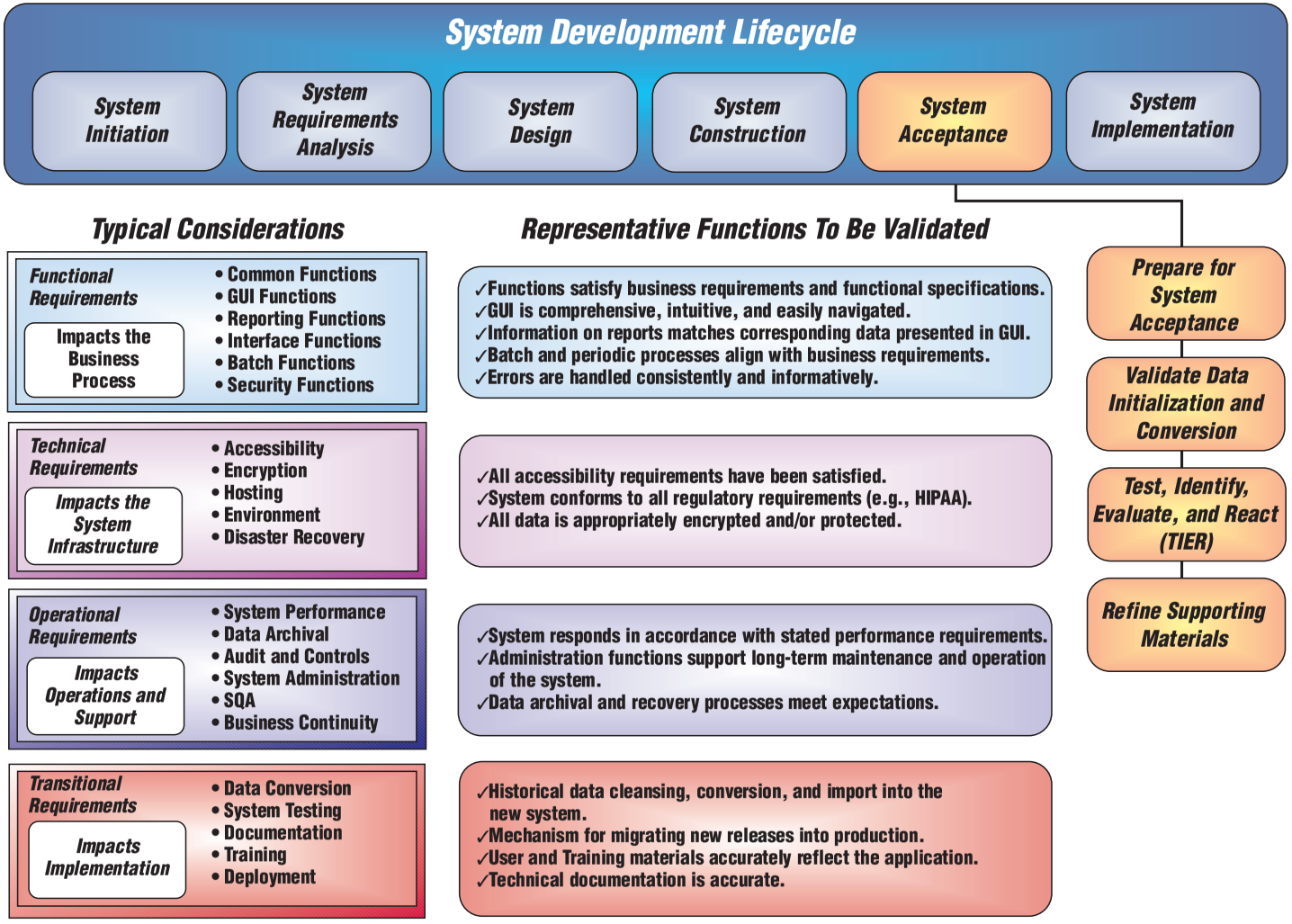
tives for validation. For all other defects, the agreed-to resolution should be communicated to all parties.

The key to successful completion of System Acceptance is the clear definition of go/no-go criteria that can be used to define the set of circumstances that would preclude placing the application into production. Should a "show stopper" be identified in these final days of testing, the Project Team must estimate and plan the appropriate corrective actions and retesting needed to resolve the problem, and then adjust the testing schedule accordingly using the standard Project Change procedures. However, if the list of issues at the end of acceptance testing contains only low priority, low impact modifications, (i.e., those that do not significantly inhibit the use of the application), testing can be considered complete. At this point, the project should progress to the next phase, with all remaining issues addressed through the application support mechanisms.

Deliverable

- **Acceptance Test Results** – A comprehensive set of completed test plans identifying all acceptance tests that were performed, along with the detailed outcomes of these tests, the list of defects identified as a result of these tests, and the results of any subsequent retests. These test results are contained in the Acceptance Test Plan section of the Technical Specifications.

Figure 5-3 System Acceptance Considerations



5.4 REFINE SUPPORTING MATERIALS

Purpose

Refine Supporting Materials ensures that all materials relating to the new application are kept up-to-date with any changes that may be introduced during System Acceptance.

Description

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Technical Writer
- SQA Lead
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Stakeholders

Despite the best efforts of the Project Team throughout the earlier phases of the lifecycle, it is common for acceptance testing activities to uncover issues that require changes to the application. In the best cases, these may be nothing more than small cosmetic changes. In extreme cases, defects detected during testing could result in major subsystems of the application being redesigned and rewritten. Regard- less of the situation, all supporting materials, (both Consumer- and Technical Support-oriented), should be reviewed to make sure that they still accurately reflect the system that will be deployed in System Implementation.

Deliverables

- **Revised User/Training Materials** – An updated set of materials aimed at assisting Consumers with the use and operation of the application, reflecting any changes that were introduced as a result of acceptance testing efforts.

- **Revised Technical Documentation** – A corresponding set of updated technical materials, again reflecting any changes introduced as a result of acceptance testing efforts and defining aspects of the application that will be useful to those individuals responsible for on-going system maintenance.

Measurements of Success

The ultimate measurement of success for System Acceptance is the agreement by the Customer to move the system into production.

Meanwhile, the Project Manager can still assess how successfully the project is proceeding by utilizing the measurement criteria outlined below. More than one "No" answer indicates a serious risk to the eventual success of the project.

Figure 5-4

Process	Measurements of Success	Yes	No
Prepare for System Acceptance	Do you have the commitment from Customer Decision-Makers to make the right people available to the extent necessary for the duration of Acceptance activities?		
	Does your testing community agree that they are adequately prepared for the Acceptance activities?		
	Does everyone have access to and the correct security level for the system?		
Validate Data Initialization and Conversion	Can you say with confidence when each outstanding data initialization and conversion defect in the log will be corrected?		
	Do your Customers agree with your assessment?		
Test, Identify, Evaluate, and React (TIER)	Can the developers fixing the defects determine, based on the defect log and test results, what the problem scenario was and what outcome was expected vs. what was experienced?		
	Are retesting efforts demonstrating that reported defects are being resolved with new releases, and that the same issues are not being reported from iteration to iteration?		
Refine User and Training Materials	Have you made changes to the user/training materials as a result of your experiences in user training and acceptance testing in this phase?		
	Have you made changes to the Technical Documentation as a result of its review by a representative of the group that will assume responsibility for the system once it's deployed?		

Phase Risks / Ways to Avoid Pitfalls

PITFALL #1 – YOU EXPECT ME TO DO WHAT?

The long Construction cycle is finally over. The system is pretty much done. Your team knocked itself out delivering what was promised, on time, within budget. You can hardly curb your enthusiasm as you call the Customer Decision-Maker to invite his cohort to spend a few weeks in the trenches slugging it out with the remaining system bugs. Curiously, all you get is dead silence, followed by a string of strangely unintelligible exclamations. Oops!

Customers (and Consumers), especially ones not experienced with formal system acceptance activities, assume that the new system will just materialize on their desktops, free of defects and perfect in every way. They view it the same way they view shrink-wrapped software packages, and have no idea how much effort goes into getting the system to the turnkey stage. It is a great shock for them to learn that, in addition to letting the system developers know what they wanted at the beginning, they need to verify at the end that what the developers actually developed meets their expectations.

Since the acceptance activities are fairly rigorous and protracted, it behooves an astute Project Manager to set those expectations way up front. Disarm them with the intention of making sure they got what they asked for, detail for them the acceptance activities and the expected level of participation and commitment, and keep reminding them, as System Acceptance nears, of their promises of people and time.

PITFALL #2 – WHAT THEY WANT VS. WHAT THEY ASKED FOR

OK, you avoided the pitfall #1 above, and an eager and agreeable group of Customers traipsed over to your neck of the woods to try out the new system. However, the honeymoon is over real quick when they actually try out the new functions. Between System Requirements Analysis and System Acceptance, time has passed, things changed and people moved around, and now nobody remembers who wanted what and why; they just know that what they see is not something they want to get.

One of the hardest things to manage during the system development lifecycle is expectations. Keeping a good audit trail should help. How good were your deliverable definitions? How tight were your acceptance criteria? Were those deliverable approval signatures written in blood – or water?

The answers to these questions spell a difference between orderly change control, and unmitigated disaster.

PITFALL #3 – FLOATING THE GARBAGE DOWNSTREAM

Finally, you avoided both of the above pitfalls, and the Customer Representatives are oohing and aahing about the system design … until they actually try to DO something. Then, all heck breaks loose: the navigation is off, the business logic is faulty, the system crashes, and the dreaded hourglass just keeps flipping over and over and over and over… endlessly…until the Customers see red behind the Blue Screen of Death. It is obvious that the system was not tested properly, and the Customers naturally resent it. Nasty rumors begin to spread, and instead of the welcome mat, the Consumers ready tar and feathers for the system deployment ceremonies.

In the heat of the construction homestretch, the temptation is to take short-cuts assuming any problems can be fixed downstream: cutting corners on software quality assurance at the unit test level, hoping to make it up during integration testing; skipping functionality reviews, hoping that the Customers will catch the errors during acceptance testing; even short-shrifting the initial training, hoping to make up for it during Implementation.

The problem is, there is never enough time in subsequent phases either. Plus, the expectations have not been set up. So if you float the consequences of bad decisions downstream, you'll just have a bigger pile of trash to deal with, instead of unfurling your sails and parading across the finish line.

PITFALL #4 – "IT'S TOO LATE, BABY!"

Another consequence of trying to short-cut the process upstream and hoping to make it up downstream is that a point comes when it's too late to address some issues. In System Acceptance, it's too late to fix system performance problems. It's too late to correct data conversion routines. It's too late to redefine core functionality, and it may even be too late to introduce minimal business process changes.

Problems like that cannot be fixed in this phase. If you can't avoid them, what you need to do is go back, and loop the life-cycle over. For data conversion problems, you probably need to go back to Construction. For performance problems, to Design. And as for problems with core functionality (with apologies to Carole King) – "Something inside has died" and you'd better push the old STOP button and rewind to the beginning; then, maybe, "there will be good times again."

PITFALL #5 – PLAYING THE BLAME GAME

When the Customer is unhappy with the new system, and is threatening to "pull the plug" on acceptance, the temptation on the part of many members of the team is to play the blame game: it's someone else's fault! The database administrators blame the network people; the network folks blame the system developers; and the programmers blame the ultimate catch-all: user error. The problem is, among all the finger-pointing, the real problem goes unsolved.

As Project Manager, your job is to keep the team together and remind people that only by pulling together will they prevail. Insist on everyone acting as a team, concentrating on solutions rather than problems, and helping each other rather than blaming the other guy.

Frequently Asked Questions

What do I do when my Project Budget does not include a QA function?

The simple fact is that quality is an indispensable part of any effort (from both project management and system development perspectives). Building a product without quality controls is wrought with risk: it will not satisfy your Customer, and will reflect poorly on your reputation.

Assuming that you cannot do a change control to add a QA function to your Project Budget, the good news is that, in a pinch, you can do without a separate QA function by incorporating quality assurance into every procedure and task, and taking on quality control responsibilities yourself.

You need to incorporate rigorous review cycles into production and acceptance of every deliverable, by using peer review mechanisms as well as inviting external SME's. As is stated in the text above, "It is more important *that* the reviews be done than *how* they are done." Sometimes even having non-technical independent observers sitting in on reviews brings extra gravity and rigor to the proceedings. Another trick is getting the Customers even more closely involved in reviewing works in progress and providing feedback.

Finally, you will need to roll up your sleeves and personally check out test scripts and acceptance procedures, and join in the testing activities – not necessarily to do any significant testing yourself, but to ensure that it gets done thoroughly and correctly.

Who should be doing the testing? I can't trust the developers to check their own work! Can I?

There are organizations where the testing function is separated into its own business unit. There are other organizations that have a separate QA function, but its members join the Project Teams at certain parts of the lifecycle, and perform testing on site. Finally, there are organizations that incorporate quality in everything they do, and thus have no separate QA function.

Each approach has its own pluses and minuses, but the important concepts are:

1. Test plans and scripts need to be developed before any coding is done
2. Test scripts need to be executed faithfully, and the results communicated immediately to the developers
3. System development should not proceed until the defects have been corrected
4. The same defects in different testing cycles point to a serious problem that has to be resolved before any further work is done

As far as the developers are concerned, sure you can trust them! To check each other's work, that is.

Who owns development of acceptance test plans? Who should the Project Manager enlist in their development?

Since Customer Representatives execute acceptance test plans, they ought to participate in their development. They need to understand the process, and be comfortable with its documentation. Plus, they probably can think up testing scenarios that the developers would never imagine!

The ownership of the process, though, still rests with the Project Team, and ultimately with the Project Manager.

6 SYSTEM IMPLEMENTATION

Purpose

The purpose of **System Implementation** can be summarized as follows: making the new system available to a prepared set of users (the deployment), and positioning on-going support and maintenance of the system within the Performing Organization (the transition). At a finer level of detail, deploying the system consists of executing all steps necessary to educate the Consumers on the use of the new system, placing the newly developed system into production, confirming that all data required at the start of operations is available and accurate, and validating that business functions that interact with the system are functioning properly. Transitioning the system support responsibilities involves changing from a system *development* to a system *support and maintenance* mode of operation, with ownership of the new system moving from the Project Team to the Performing Organization.

A key difference between System Implementation and all other phases of the lifecycle is that all project activities up to this point have been performed in safe, protected, and secure environments, where project issues that arise have little or no impact on day-to-day business operations. Once the system goes live, however, this is no longer the case. Any miscues at this point will almost certainly translate into direct operational and/or financial impacts on the Performing Organization. It is through the careful planning, execution, and management of System Implementation activities that the Project Team can minimize the likelihood of these occurrences, and determine appropriate contingency plans in the event of a problem.

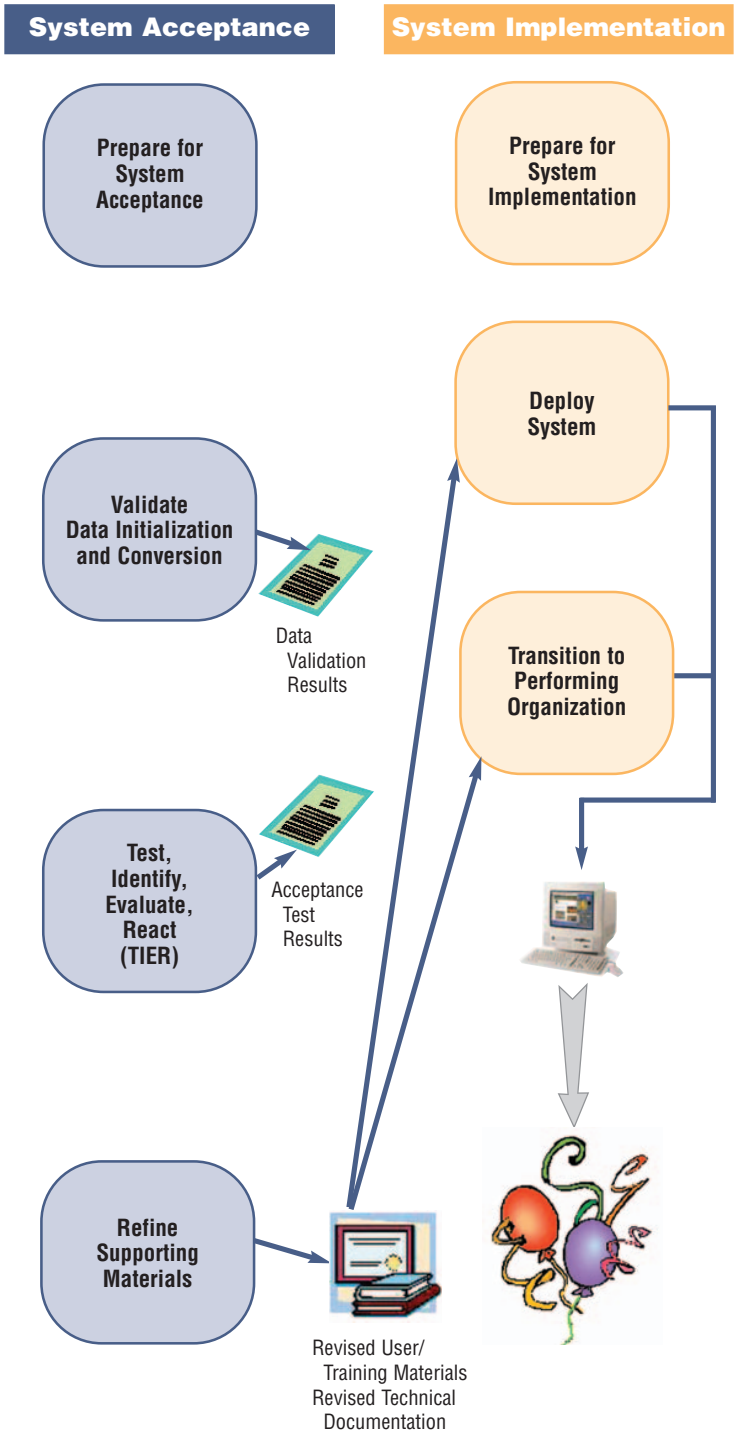
List of Processes

This phase consists of the following processes:

- **Prepare for System Implementation,** where all steps needed in advance of actually deploying the application are performed, including preparation of both the production environment and the Consumer communities.
- **Deploy System,** where the full deployment plan, initially developed during System Design and evolved throughout subsequent lifecycle phases, is executed and validated.
- **Transition to Performing Organization,** where responsibility for and ownership of the application are transitioned from the Project Team to the unit in the Performing Organization that will provide system support and maintenance.

The following chart illustrates all of the processes and deliverables of this phase in the context of the system development lifecycle.

Figure 6-1


System Acceptance
System Implementation
Prepare for System Acceptance
Prepare for System Implementation
Deploy System
Validate Data Initialization and Conversion
Data Validation Results
Transition to Performing Organization
Test, Identify, Evaluate, React (TIER)
Acceptance Test Results
Refine Supporting Materials
Revised User/ Training Materials
Revised Technical Documentation

List of Roles

The following roles are involved in carrying out the processes of this phase. Detailed descriptions of these roles can be found in the Introductions to Sections I and III.

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- Application Developers
- Software Quality Assurance (SQA) Lead
- Technical Services (HW/SW, LAN/WAN, TelCom)
- Information Security Officer (ISO)
- Technical Support (Help Desk, Documentation, Trainers)
- Customer Decision-Maker
- Customer Representative
- Consumer
- Performing Organization
- Stakeholders

List of Deliverables

The following table lists all System Implementation processes, some techniques available for use in executing these processes, and process outcomes and deliverables.

Figure 6-2

Processes	Techniques	Process Deliverables *(Outcomes)*
Prepare for System Implementation	Interviews Distribution of Materials Coordination of Training Logistics	*Established Team and Environment for System Implementation*
Deploy System	Training Sessions Manual Business Operations Parallel Operations	*Migrated and Initialized Data* *Operational System*
Transition to Performing Organization	Training Sessions Phased Ownership	*Ownership of System by Performing Organization*

6.1 PREPARE FOR SYSTEM IMPLEMENTATION

Purpose

The purpose of **Prepare for System Implementation** is to take all possible steps to ensure that the upcoming system deployment and transition occurs smoothly, efficiently, and flawlessly.

Description

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- SQA Lead
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Consumer
- Performing Organization
- Stakeholders

In the implementation of any new system, it is necessary to ensure that the Consumer community is best positioned to utilize the system once deployment efforts have been validated. Therefore, all necessary training activities must be scheduled and coordinated. As this training is often the first exposure to the system for many individuals, it should be conducted as professionally and competently as possible. A positive training experience is a great first step towards Customer acceptance of the system.

During System Implementation it is essential that everyone involved be absolutely synchronized with the deployment plan and with each other. Often the performance of deployment efforts impacts many of the Performing Organization's normal business operations. Examples of these impacts include:

- Consumers may experience a period of time in which the systems that they depend on to perform their jobs are temporarily unavailable to them. They may be asked to maintain detailed manual records or logs of business functions that they perform to be entered into the new system once it is operational.
- Technical Services personnel may be required to assume significant implementation responsibilities while at the same time having to continue current levels of service on other critical business systems.
- Technical Support personnel may experience unusually high volumes of support requests due to the possible disruption of day-to-day processing.

Because of these and other impacts, the communication of planned deployment activities to all parties involved in the project is critical. A smooth deployment requires strong leadership, planning, and communications. By this point in the project lifecycle, the team will have spent countless hours devising and refining the steps to be followed. During this preparation process the Project Manager must verify that all conditions that must be met prior to initiating deployment activities have been met, and that the final 'green light' is on for the team to proceed.

More than at any other point in the project, the Project Manager must plan for failure, and must have a defined set of contingency plans to be executed in the event of a problem encountered during deployment. Stakeholders and all key decision-makers must clearly understand and agree to the various "go/no go" criteria by which decisions will be made whether or not to proceed with the deployment. In the event of a failure, time lost as a result of an ill-defined course of action can be costly not only in terms of Project Budget, but equally important, in terms of Customer and Consumer confidence.

The final process within the System Development Lifecycle is to transition ownership of the system support responsibilities to the Performing Organization. In order for there to be an efficient and effective transition, the Project Manager should make sure that all involved parties are aware of the transition plan, the timing of the various transition activities, and their role in its execution.

Due to the number of project participants in this phase of the SDLC, many of the necessary conditions and activities may be beyond the direct control of the Project Manager. Consequently, all Project Team members with roles in the implementation efforts must understand the plan, acknowledge their responsibilities, recognize the extent to which other implementation efforts are dependent upon them, and confirm their commitment.

6.2 DEPLOY SYSTEM

Purpose

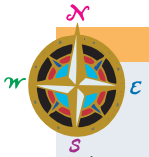
The purpose of the **Deploy System** process is to perform all activities required to successfully install the new system and make it available to the Consumers.

Description

Deploying the system is the culmination of all prior efforts – where all of the meetings, planning sessions, deliverable reviews, prototypes, development, and testing pay off in the delivery of the final system. It is also the point in the project that often requires the most coordination, due to the breadth and variety of activities that must be performed. Depending upon the complexity of the system being implemented, it may impact technical, operational, and cultural aspects of the organization. A representative sample of high-level activities might include the installation of new hardware, increased network capabilities, deployment and configuration of the new system software, a training and awareness campaign, activation of new job titles and responsibilities, and a completely new operational support structure aimed at providing Consumer-oriented assistance during the hours that the new system is available for use (to name a few). Whatever the realm of activities related to the new system, their impacts should be addressed in the Organizational Change Management Plan, while specific deployment activities should all be encompassed in the Project Implementation and Transition Plan, (both created during the Project Planning phase of the Project Management Lifecycle.)

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- SQA Lead
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Consumer
- Performing Organization
- Stakeholders



Depending upon the environment or the type of system being implemented, this phase may also warrant additional activities including 'sunsetting' (retiring) any related legacy systems, executing parallel runs, and managing external communications.

All Consumer training should be performed prior to physically migrating the system to the production environment. This will enable the Consumers to begin to familiarize themselves with the system, and will help to establish their expectations regarding what the system will and will not do.

When it comes to training, sometimes the timing of the training can be as important as the content. Conducting the training after the system has been rolled out to the Consumers may cause them to form poor perceptions of the system, simply due to the difficulties associated with an unnecessarily lengthy learning curve. Similarly, holding the training sessions too far in advance of the deployment presents Consumers with the challenge of having to recall what was taught, again leading to possible frustration and unhappiness with the system.

The sequencing of deployment activities is just as important as it was with previous testing activities. This sequence of events should be encompassed in the Deployment and Transition Plan section of the Technical Specification, and will address and prioritize any necessary training activities, set-up activities needed to prepare the production environment (desktop, LAN, servers, data center, etc.), and data conversion and validation activities. This deployment plan will also define the steps for physically migrating the system and any associated utilities to production, and for validating the accuracy and completeness of this migration after these steps have been performed. During deployment, Project Team members may often be required to work extra hours to meet aggressive timeframes, or additional staff may be brought in temporarily to assist with large data entry efforts. Proper planning and sequencing of the deployment can help to minimize these situations, and reduce the chance of any missteps that could result in having to restart the deployment process, or lengthen the implementation schedule.

As the system is enabled, and the Project Team validates that the application is performing to expectations, there may be times when certain system functions seem suspect. One of the challenges most frequently faced by Project Teams is to determine the root cause of potential issues. Discrepancies that exist within the data could be due to defects in the application's business logic, or could be the result of data that was improperly migrated into the system. Similarly, the inability of a Consumer to access specific features of the system could be caused by improperly configured hardware, or incorrectly

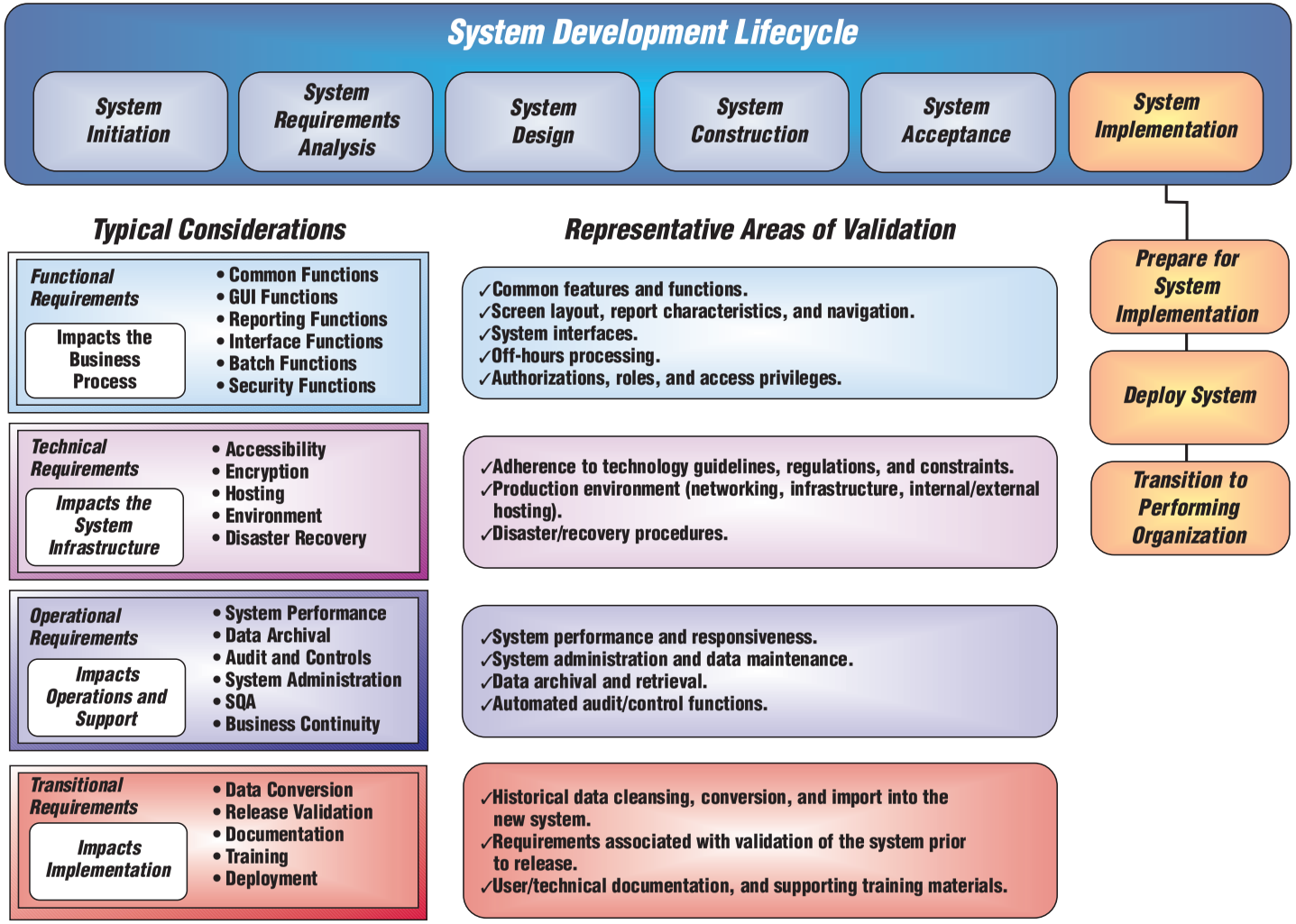
established security privileges. To minimize confusion and reduce the opportunity for such issues to surface, every attempt should be made to immediately validate each step of the deployment as it is performed.

Ideally, there will be no aspect of the implementation that was not previously tested during System Acceptance. Whether or not this is true, there is always the possibility that routines or utilities that worked properly in one environment may not work identically in another. With this in mind, the Project Team should always validate the success of each step of the deployment, and wherever possible, should take appropriate steps to enable the team to "fall back" to a prior state should the severity of a problem warrant such an action.

Additionally, the Project Manager should be aware that not everyone is open or receptive to change. As a system is rolled out to its target audience, the team must remain keenly attentive to how it is perceived. The fact that functions that were present in the legacy system no longer exist or work differently may cause some Consumers to see the new system negatively. And while the new system may provide overall benefits to the business or agency, those benefits may come at the expense of additional work responsibilities to some of the individuals who interact with the system (e.g., the new system may require the entry of additional data that was not previously required). By understanding some of the dynamics behind how the system is being received, the Project Team may be better able to identify or publicize some of the benefits that the system provides. A well-defined Organizational Change Management Plan should have anticipated and addressed these issues.

One effective way to gauge the use and acceptance of the system is for the Project Team to maintain open communications channels with the Technical Support or Help Desk operation, if one exists. This will provide a broader view of potential issues or suggestions that can then be addressed proactively.

Figure 6-3 System Implementaion Considerations



6.3 TRANSITION TO PERFORMING ORGANIZATION

Purpose

The purpose of the **Transition to Performing Organization** process is to successfully prepare the Performing Organization to assume responsibility for maintaining and supporting the new application.

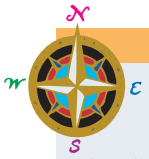
Description

In many organizations, the team of individuals responsible for the long-term support and maintenance of a system is different from the team initially responsible for designing and developing the application. Often, the two teams include a comparable set of technical skills. The responsibilities associated with supporting an operational system, however, are different from those associated with new development.

Roles

- Project Manager
- Project Sponsor
- Business Analyst
- Data/Process Modeler
- Technical Lead/Architect
- SQA Lead
- Technical Services
- Information Security Officer
- Technical Support
- Customer Decision-Maker
- Customer Representative
- Consumer
- Performing Organization
- Stakeholders

In order to effect this shift of responsibilities, the Project Team must provide those responsible for system support in the Performing Organization with a combination of technical documentation, training, and hands-on assistance to enable them to provide an acceptable level of operational support to the Consumers. This system transition is one element (albeit a major one) of the overall Project Implementation and Transition Plan, developed as part of the PM Lifecycle. The Project Manager should review the transition plan to confirm that all defined actions have been successfully completed.



One common approach to successfully transitioning support responsibilities is to implement a phased transition schedule that gradually shifts increasing ownership of the system to the Performing Organization. Early phases would have the Performing Organization's support and maintenance team primarily observing the Project Team as part of knowledge transfer, while later phases would have the support team acting as the first line of response. The exact phases and their timing should be determined as part of transition planning earlier in the lifecycle. Regardless of the approach selected, communication of the overall plan and responsibilities needs to be clear so that System Implementation can be brought to a clean and clear end.

Measurements of Success

System Implementation serves as its own Measurement of Success; indeed, a smooth System Implementation culminates – and validates – the entire system development effort.

Nevertheless, even before the final turnover, the Project Manager can utilize the measurement criteria below to assess how successfully the implementation is proceeding. More than one "No" answer indicates a serious risk to the success of this phase – and the entire project.

Figure 6-4

Process	Measurements of Success	Yes	No
Prepare for System Implementation	Has anyone verified that every Consumer has the right level of system access and security?		
	Is there a checklist of all system components that can be used to verify that all the right versions of all components of the system are in the production environment?		
	Do the managers of Technical Services and Technical Support agree with your estimate of extra work for their units associated with new system deployment?		
Deploy System	Do your team members agree that their part of the effort as outlined in the Project Implementation and Transition Plan is reasonable and achievable?		
	Do the training evaluation forms filled out by Consumers and Customers being trained in the new system reflect scores equal or higher to those anticipated in the Project Implementation and Transition Plan?		
	Have you had to "freeze" or "fall back" in system deployment activities no more than originally anticipated in the deployment plan?		
	Is the volume of support calls within the range originally anticipated in the deployment plan?		
Transition to Performing Organization	Has the Performing Organization agreed to transition all of the remaining defects along with the system itself?		

Phase Risks / Ways to Avoid Pitfalls

PITFALL #1 – DAMN THE TORPEDOES, FULL SPEED AHEAD!

Admiral David "Old Salamander" Farragut may have won the battle of Mobile Bay in 1864 with that command, but for a typical Project Manager, a planned "freeze point" should serve far better when the first mine explodes under the new system.

During the course of System Implementation, the Project Manager should have many points where the process can be frozen while the minesweepers fan out or a hole is patched. Think of it as a space shuttle countdown – NASA has frozen the clock with as little as 31 seconds before launch when the conditions warrant and a problem was discovered.

Having multiple pre-planned go/no go points during Implementation will spare you many grey hairs and sleepless nights.

PITFALL #2 – ABANDON SHIP! WE MESSED UP PRODUCTION!

The more extensive and complex the system, the better the chance that something will go wrong in production, no matter how well the System Acceptance phase went. That's why it behooves the Project Manager not only to execute a comprehensive check of the entire production environment EVERY time anything is moved to production, but also to have an orderly fall-back plan for restoring production to a workable condition when – not if – something goes wrong.

Some error conditions are obvious and unmistakable – wrong heading on a screen, an improperly calculated total on a report; others are insidious in their perniciousness – a database update mechanism that deletes rows infrequently and at random, or miscalculates results by a small fraction. Those conditions may persist for days before being detected, and present insurmountable challenges in absence of a deliberate plan for retreat.

Save snapshots of the database; concatenate transactions; mothball but do not discard older versions of application code. Be ready to roll back or to jump back and roll forward – as long as you are not rolling off the deck of a sinking ship.

PITFALL #3 – IT'S ALL MY FAULT!

There is enough blame to go around in a typical System Implementation scenario. Something gets forgotten, something does not work right, something happens that is not planned for… and a good first step in fixing the problem is acknowledging the responsibility for it. However, sometimes the fault is not yours – even if someone is convinced it is.

Take, for example, a Consumer's reaction to the new system, especially if said Consumer was not directly involved in System Acceptance. It is possible, indeed likely, that some feature of the new system will be at odds with what the Consumer thinks it ought to be. After all, it was requested by somebody else, and somebody else again built it. So a vocal Consumer will complain that the new system is wrong.

This situation is dangerous on two fronts: first, if accepted on its face value, it may mean rework, delays or worse; second, if not handled correctly, it may taint the perception of the new system and may lead to more complaints.

This is another case where solid, signed documentation really pays off. Prove that the functionality works just as it was requested. Enlist the help of the Project Sponsor, the Customer Decision-Makers, and any other "persuasive peers" (who are most likely just as anxious to have the system well received as you are), to explain the rationale behind design decisions and the process for change. And don't accept any more blame than is properly yours.

PITFALL #4 – THE IMPLEMENTATION THAT NEVER ENDS

There is a song that never ends, it just goes on and on, my friends… at least until you stop singing it (probably because people are throwing things at you). But what if you are stuck in the Implementation phase that just does not seem to end? As soon as you fix all the technical problems, a Consumer reports a new bug, and you go through the cycle of fixing and testing and moving and checking, and then another Customer requests an urgent change, and the implementation cycle starts all over again, and then you encounter another technical issue, and it just goes on and on and on…

Somewhere – preferably sooner rather than later – you have to cut the cord, cross the Rubicon, make like a tree and leave well enough alone.

It certainly helps if you have a planned turnover procedure and a solid Project Transition Plan. It is also beneficial to have a Project Sponsor who understands the difference between system development and system support. And most of all, it is necessary to have courage and resolve to say "Basta!" "Finito!" and "Arrivederci!" (and not necessarily in Italian, either.)

Frequently Asked Questions

What is a pilot? How do we do it? How do we move out of the pilot phase?

In system development parlance, a "pilot" is one of the techniques for deploying the system. Another technique is called "phased implementation," although both terms are sometimes used interchangeably.

There are four main ways to roll out the new system. One – you can do it all at once, deploying all parts of the system to all the Consumers in one fell swoop. Two – you can deploy it piecemeal, releasing some of the system today, a little bit more tomorrow, taking it easy, rolling it out one part at a time. (That's what we'll call "phased implementation.") Three – you can release the whole system in one shot, but only to a small group of friendly users. Once you verify that your test community survived the experience, you roll the system to another group, moving up the chain until you dare to expose your creation to your harshest critics. That's what we'll call a "pilot". Finally, for very large and mission critical systems, you can do phased implementation in a pilot mode – roll out parts of the system to small groups.

Each approach has its pluses and minuses. Specifically for the pilot, the advantages are lessened exposure and an extra opportunity to test the system before releasing it to the world. The great disadvantages are having to maintain and coordinate two parallel processes, stretching out the deployment, and tying up Project Team resources for an extra long time.

If you do decide on a pilot, make sure the pilot group represents if not all, then at least the lion's share of business requirements. And once you document that the system can handle them adequately – move on. It is very important to limit your actions to addressing legitimate system bugs only; if you start tweaking and enhancing and adding functionality, you will never get out of the pilot phase.

Does the system need to be perfect before deployment?

"Perfect" according to whom? What may be acceptable for one Customer may not be good enough for another; and unless you involve your whole user base in accepting a fully functional system, you will not know whether your system is "perfect"; and by then – guess what – you've deployed it!

If you had a good representative cross-section of Customers and Consumers doing your System Acceptance, if you had cogent acceptance testing plans for them, if the Customer Decision-Makers signed off on acceptance test results, and if your team is confident in your deployment plan – by all means, take the leap, and release your baby into the wild.