

## 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

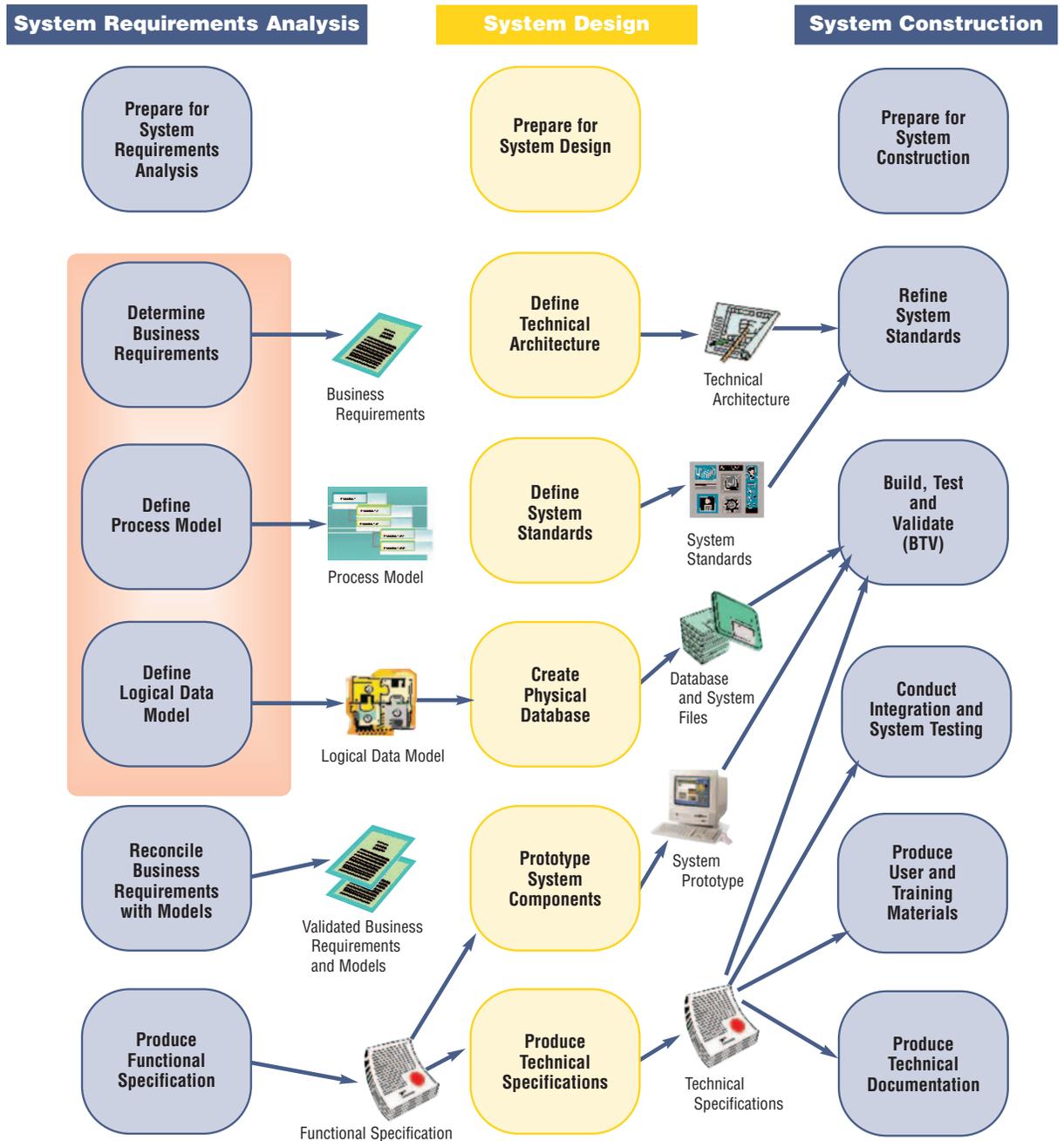
This phase consists of the following processes:

- ◆ **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



### 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**

<b>Processes</b>	<b>Techniques</b>	<b>Process Deliverables (Outcomes)</b>
Prepare for System Design	Interviews Site Walk-throughs	<i>Established Team and Environment for System Design</i>
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.

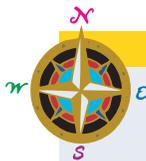
The initiation of a new project phase is also the right time to assess the training needs of the existing team. Please refer to Section I, Project Planning, for a detailed approach to developing your Project Team.

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)

<p style="text-align: center;"><b>Technical Architecture</b></p> <p style="text-align: center;"><b>TABLE OF CONTENTS</b></p> <div style="border: 1px solid black; padding: 5px; margin: 10px 0;"><p><i>The <b>Table of Contents</b> should be at least two levels deep.</i></p></div> <p><b>1.0 DOCUMENT SCOPE</b></p> <div style="border: 1px solid black; padding: 10px;"><p><i><b>Document Scope</b> describes the context and the goals of this document in a narrative.</i></p><p><i>Example:</i></p><p><i>This document describes the Technical Architecture of the &lt;XYZ&gt; System that satisfies business requirements as documented in the Business Requirements Document, &lt;Date&gt;, and implements the functionality and satisfies technical, operational and transitional requirements described in the Functional Specification, &lt;Date&gt;.</i></p><p><i>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.</i></p><p><i>This document will also:</i></p><ul style="list-style-type: none"><li>● <i>Identify and explain the risks inherent in this Technical Architecture;</i></li><li>● <i>Define baseline sizing, archiving and performance requirements;</i></li><li>● <i>Identify the hardware and software specifications for the Development, Testing, QA and Production environments;</i></li><li>● <i>Define procedures for both data and code migration among the environments.</i></li></ul><p><i>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.</i></p></div>
---

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.

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.

#### 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 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)

<h2>System Standards</h2>	
<b>2.0 MODULE DEVELOPMENT STANDARDS</b>	
<b>2.1 Graphical User Interface</b>	
<i>Graphical User Interface standards address such areas as:</i>	
<ul style="list-style-type: none"><li>● Screen Design (Resolution, layout, menus, tabs, messages, and other screen components)</li><li>● Visual Design (Colors, fonts, icons, buttons, symbols, system pointer, tabs and other visual components)</li><li>● Operational Design (GUI component behavior, operational consistency, application response (instructions and messages), and other operational behaviors)</li><li>● Usability Design (simplicity, clarity, fault tolerance, system feedback, and other usability attributes)</li></ul>	
<b>2.2 Reporting</b>	
<i>Reporting standards address such areas as:</i>	
<ul style="list-style-type: none"><li>● Report Design (Layout, Mandatory fields, Identifying fields, Fonts, Colors, Graphical elements)</li><li>● Numerical Data Representation (Formats, Totals, Rounding)</li><li>● Usability Design (Simplicity, clarity, messages, explanations, user-controlled variables)</li></ul>	
<b>2.3 Application Navigation</b>	
<i>Application Navigation standards address such areas as:</i>	
<ul style="list-style-type: none"><li>● Menu Structure (Levels, access, behavior, verbosity)</li><li>● Selective Navigation (Beginner, Intermediate and Expert navigation paths)</li><li>● Navigation Aids (Navigation buttons, keyboard commands, navigational messages)</li><li>● Keyboard Shortcuts (Special key combinations for frequently-used operations)</li></ul>	
<b>2.4 Error Prevention and Correction</b>	
<i>Error Prevention and Correction standards address such areas as:</i>	
<ul style="list-style-type: none"><li>● Input Guidance (Variable field protection, value selection, informational messages, directional messages, input fields position/presentation/behavior)</li><li>● Input Validation (Validation order, edit checking, alternative presentation, choice verification)</li><li>● Error Handling (Routines, messages, responses)</li></ul>	
<b>2.5 Programming</b>	
<i>Programming standards address such areas as:</i>	
<ul style="list-style-type: none"><li>● Coding standards (Organization, structure, style, consistency)</li><li>● Documentation standards (Placement, verbosity, style)</li><li>● Development environment elements (Objects, packages, methods, variables, parameters, etc.)</li><li>● Debugging (Techniques, routines, messages)</li></ul>	

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)

<h2>System Standards</h2>	
<b>3.2 QA Environment</b>	
3.2.1 Software Management	
3.2.2 Database Management	
<b>3.3 Acceptance Environment</b>	
3.3.1 Software Management	
3.3.2 Database Management	
<p><i>For each environment necessary for system construction (<b>Development, QA and Acceptance</b>), identify <b>Software Management</b> procedures (source code version control, application version control, backup and recovery, etc.) and describe procedures and controls used for <b>Database Management</b> (data sources, migration and synchronization procedures, data backup and recovery, etc.)</i></p>	
<b>4.0 RELEASE MANAGEMENT STANDARDS</b>	
<b>4.1 Migration from Development to QA Environments</b>	
4.1.1 Software Migration	
4.1.2 Data Migration	
<b>4.2 Migration from QA to Acceptance Environments</b>	
4.2.1 Software Migration	
4.2.2 Data Migration	
<p><i><b>Release Management Standards</b> detail how source code, compiled applications, and data will be migrated among the various environments (Development, QA, and Acceptance).</i></p>	
<b>5.0 TESTING STANDARDS</b>	
<b>5.1 Unit Testing</b>	
5.1.1 Unit Testing Standards	
5.1.2 Unit Testing Tools	
<b>5.2 Integration and System Testing</b>	
5.2.1 Integration/System Testing Standards	
5.2.2 Integration/System Testing Tools	
<b>5.3 Acceptance Testing</b>	
5.3.1 Acceptance Testing Standards	
5.3.2 Acceptance Testing Tools	
<p><i>For each kind of testing performed (<b>Unit, Integration/System and Acceptance</b>), define <b>Testing Standards</b> and suggested approaches to setting up test cases and conducting tests, and identify and describe <b>Testing Tools</b> that should be utilized in that testing cycle.</i></p>	

**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.

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.

**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



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.



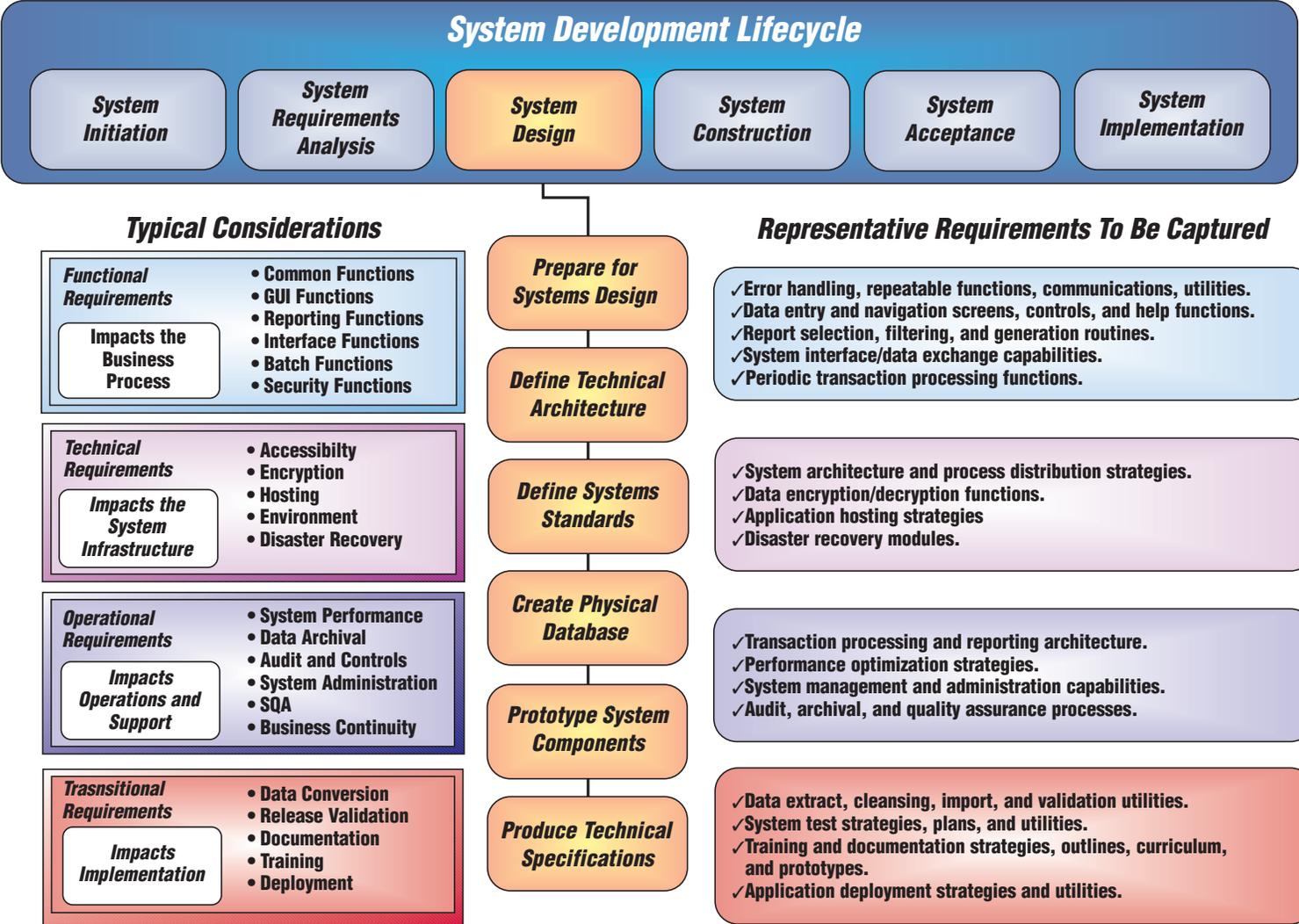
A common mistake made on many System Development projects is attempting to reduce or remove the Business Analyst function from the Project Team once the requirements have been captured. This can lead to many potential pitfalls, since it then places the burden of transforming these requirements into Technical 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-

validation 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:

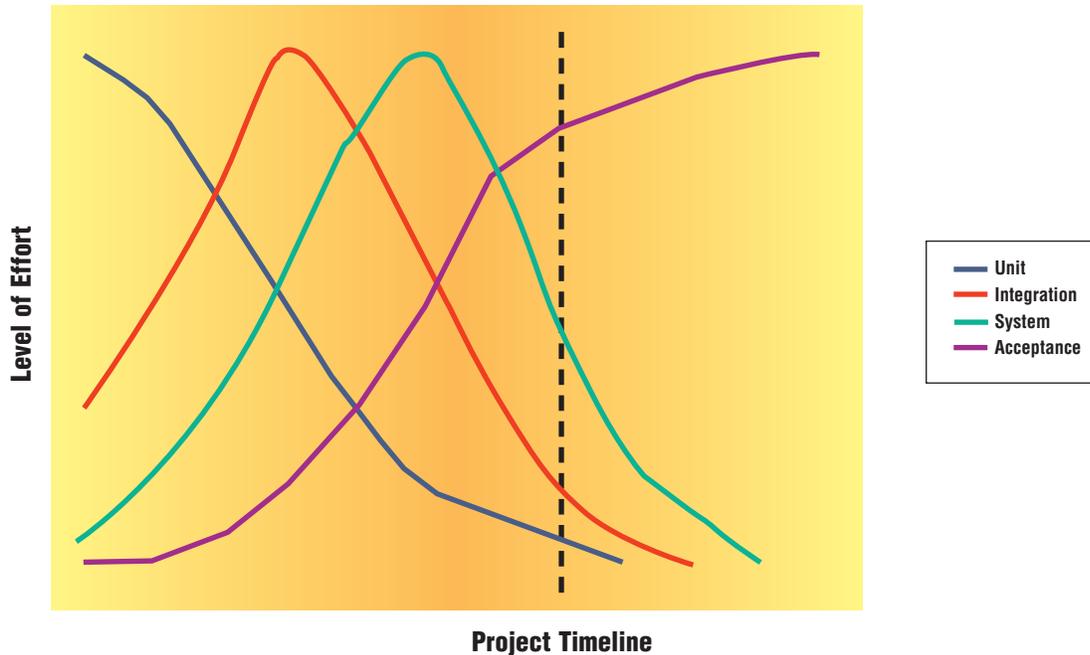
- ◆ **Unit Testing**, where individual system components are independently tested as they are developed to ensure that each logic path contained within each module performs as expected. Many tests performed during unit testing can be used for more than one module (error handling, spell checking of screens 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 be 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

<p>&lt; Name of Agency &gt;</p> <h2 style="text-align: center;">Technical Specifications</h2> <p>&lt; System Name &gt;</p>	
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)

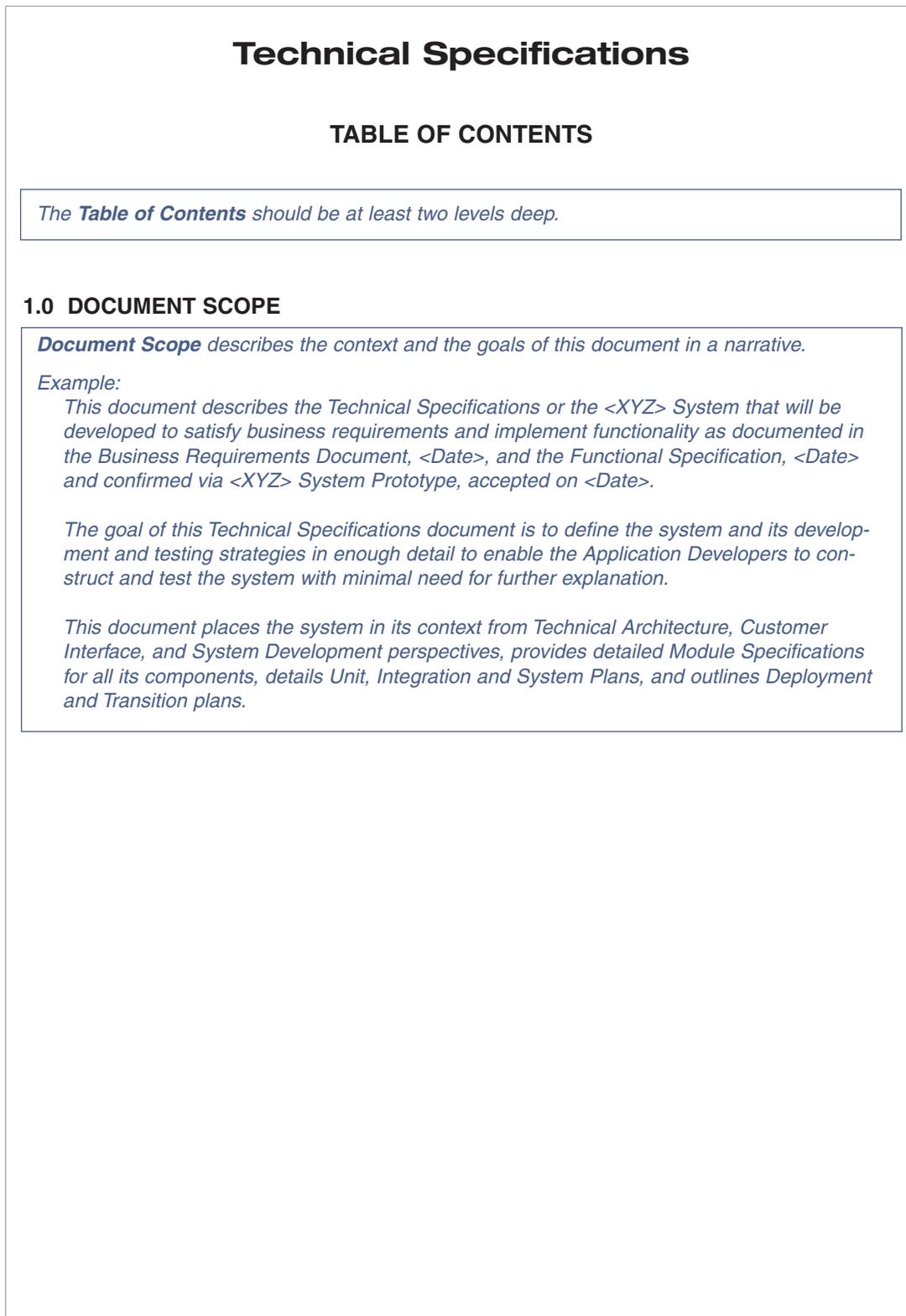


Figure 3-7 (Continued)

## Technical Specifications

### 2.0 SYSTEM ARCHITECTURE

*The **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)

<b>Technical Specifications</b>	
3.1.1.5	<i>Business Requirement(s)</i>
<i>Business Requirements</i> provides a tie-back to the Business Requirements Document.	
3.1.1.6	<i>Inputs</i>
<i>Inputs</i> details all data sources, Consumer input, etc. that will provide data to the module.	
3.1.1.7	<i>Interfaces</i>
<i>Interfaces</i> 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	<i>Security Considerations</i>
<i>Security Considerations</i> identify how the security strategy will be implemented in the module.	
3.1.1.9	<i>Logic Flow</i>
<i>Logic Flow</i> details how the business rules will be implemented by module code.	
3.1.1.10	<i>Outputs</i>
<i>Outputs</i> details all data stores, displays, etc. created or modified as a result of the module's execution.	
3.1.1.11	<i>Database Access</i>
<i>Database Access</i> 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	<i>Common Elements Used</i>
<i>Common Elements Used</i> lists all the common objects, stored procedures, etc. that will be used by the module.	
3.1.1.13	<i>Module Review Process</i>
<i>Module Review Process</i> outlines what QA procedures will be used to review the module and its test results.	
3.1.1.14	<i>Audit Tracking</i>
<i>Audit Tracking</i> 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	<i>Special Considerations</i>
<i>Special Considerations</i> allows compilation of all other Requirement, Design and/or Construction considerations that the Application Developer should be cognizant of.	
3.1.1.16	<i>Unit Test Plan</i>
<i>Unit Test Plan</i> 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)

<b>Technical Specifications</b>	
System Test Results:	
Tester:	
Name:	
Date:	Time:
Results:	
Passed: _____	Failed: _____
Justification:	
Verifier:	
Name:	
Date:	Time:
<p><b>System Test Case Number</b> allows quick reference to test case; should be based on module identification.</p> <p><b>System Test Case Name/Purpose</b> provide a brief description of the scenario being tested.</p> <p><b>Module List</b> identifies system modules included in the Packet</p> <p><b>System Test Data</b> identifies data values (or conditions) that need to be set in order to conduct the test case.</p> <p><b>Navigation</b> provides a sequence of activities that need to be performed to set up and execute the test.</p> <p><b>Expected Results</b> 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.</p> <p><b>Comments</b> provides additional considerations for the test (expected Fail conditions, etc.)</p> <p><b>System Test Results</b> allows the tester to record the results of the test.</p> <p><b>Tester</b> enters his/her <b>Name</b>, and <b>Date</b> and <b>Time</b> of the test, certifies the test as <b>Passed</b> or <b>Failed</b>, 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.</p> <p><b>Verifier</b> verifies that the System Test was conducted as described, and produced reported results.</p>	
<b>5.2 System Test Packet 2</b>	
<b>6.0 ACCEPTANCE TEST PLAN</b>	
<p><b>Acceptance Test Plan</b> details the activities to be performed in integration and system testing.</p>	
<p>Modules, groups of modules and sub-systems are organized into <b>Acceptance Test Packets</b> to facilitate Customer Representative testing of the system.</p>	
<b>6.1 Acceptance Test Packet 1</b>	
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 bird-house! 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.