4 SYSTEM CONSTRUCTION

Purpose

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

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

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

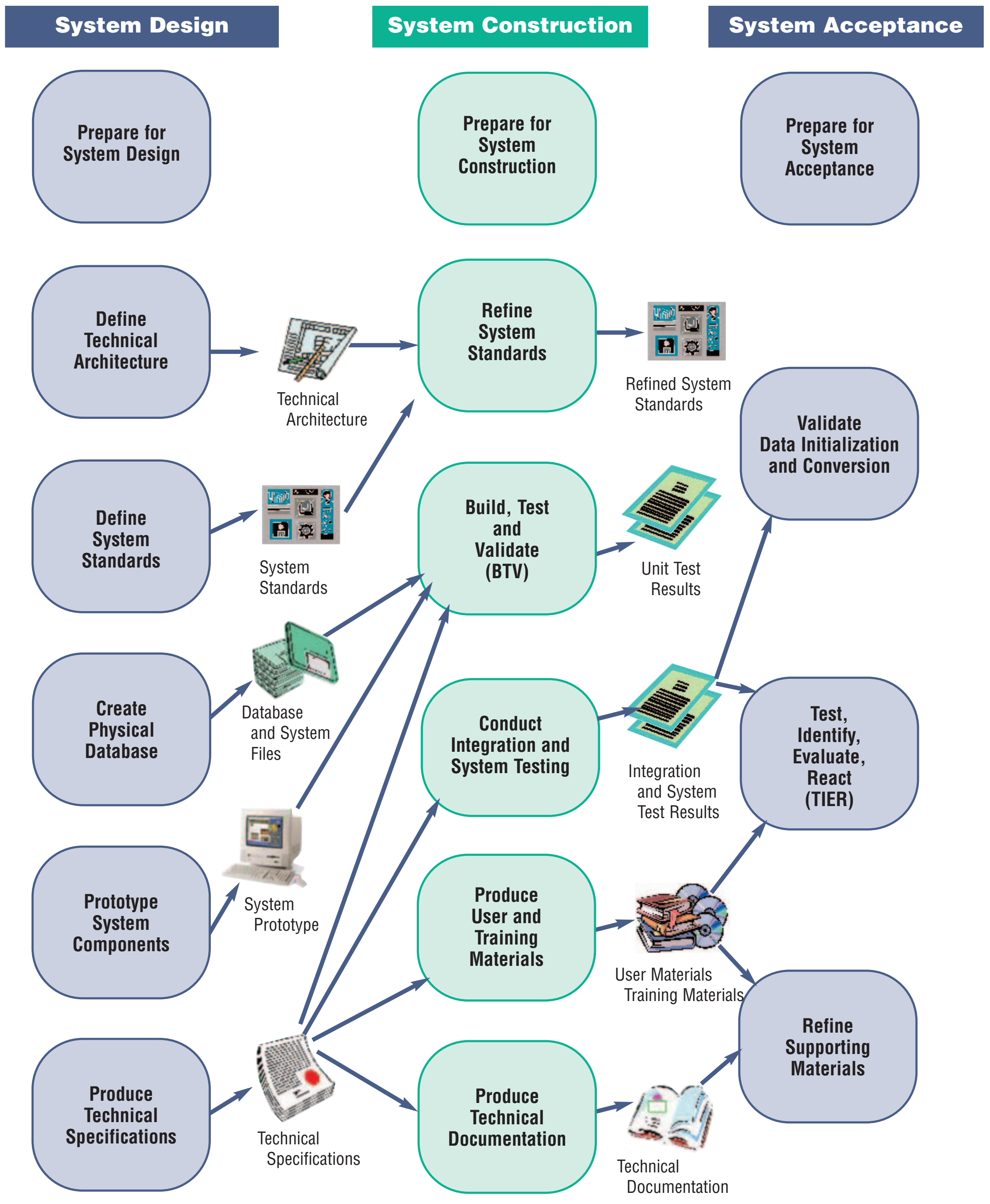
This phase consists of the following processes:

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

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

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

Figure 4-1


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

List of Roles

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

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

List of Deliverables

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

Figure 4-2

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

4.1 PREPARE FOR SYSTEM CONSTRUCTION

Purpose

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

Description

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

Roles

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

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

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

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

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

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

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

4.2 REFINE SYSTEM STANDARDS

Purpose

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

Description

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

Roles

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

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

While many Project Teams view this process as primarily focused on the refinement of technical standards that apply directly to the system being developed, the same evaluation and appraisal process can be applied to any of the processes or practices being followed by the development team. Periodic reviews and refinement of all such processes can help reduce project risks and ensure that the processes culminate in the desired results. Depending upon factors such as team size or project complexity, reviews can range from half-day facilitated sessions to a simple five-minute discussion once a month during regular team meetings. The Project Manager must determine the approach that is most appropriate for each project. It is more important that the reviews are done, than *how* they are done.

Deliverable

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

4.3 BUILD, TEST, AND VALIDATE (BTV)

Purpose

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

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

Build

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

Roles

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

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

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

Test

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

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

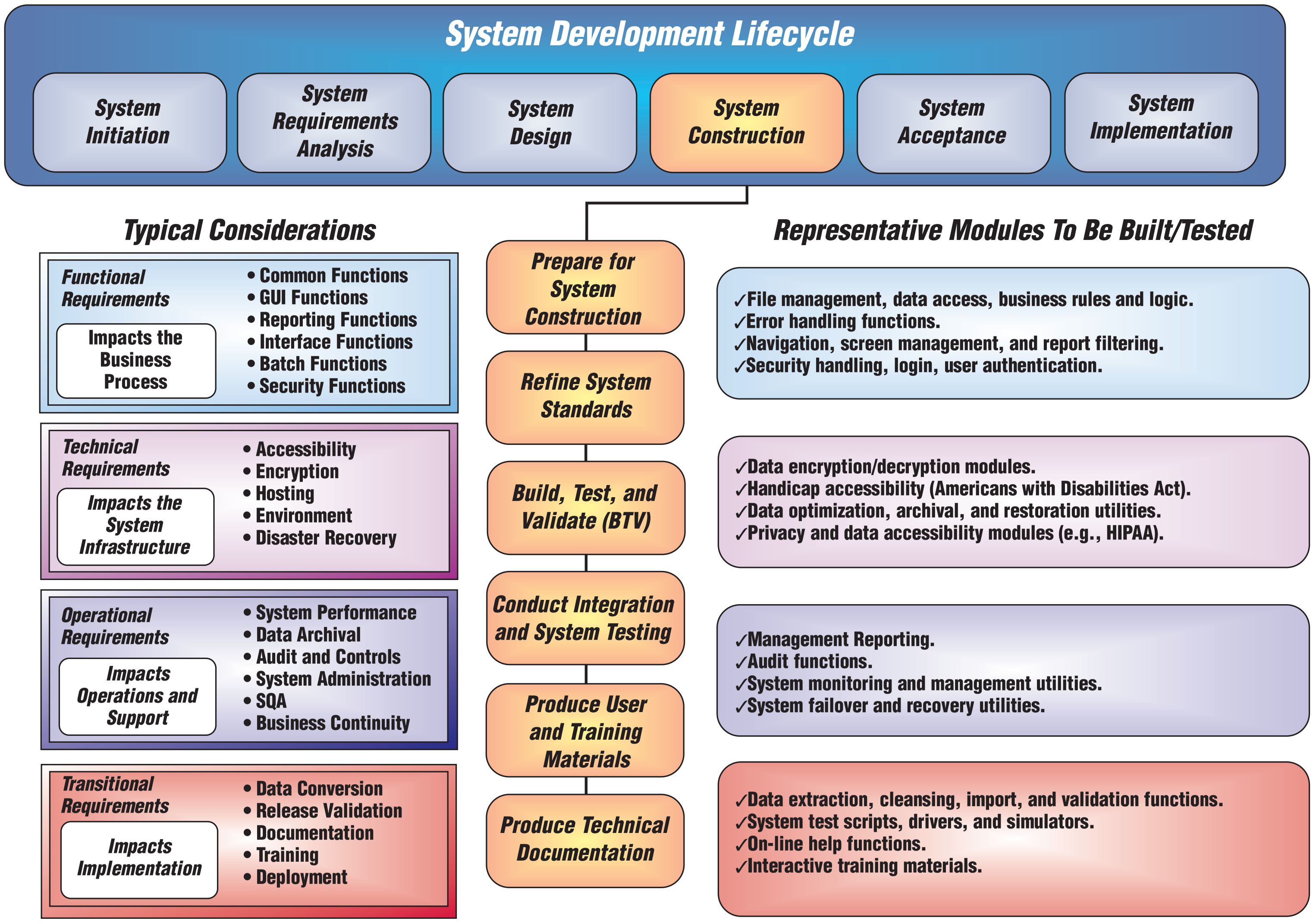
Validate

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

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

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

Figure 4-3 System Construction Considerations


System Development Lifecycle
System Initiation
System Requirements Analysis
System Design
System Construction
System Acceptance
System Implementation
Typical Considerations
Functional Requirements
Impacts the Business Process
• Common Functions
• GUI Functions
• Reporting Functions
• Interface Functions
• Batch Functions
• Security Functions
Technical Requirements
Impacts the System Infrastructure
• Accessibility
• Encryption
• Hosting
• Environment
• Disaster Recovery
Operational Requirements
Impacts Operations and Support
• System Performance
• Data Archival
• Audit and Controls
• System Administration
• SQA
• Business Continuity
Transitional Requirements
Impacts Implementation
• Data Conversion
• Release Validation
• Documentation
• Training
• Deployment
Prepare for System Construction
Refine System Standards
Build, Test, and Validate (BTV)
Conduct Integration and System Testing
Produce User and Training Materials
Produce Technical Documentation
Representative Modules To Be Built/Tested
✓File management, data access, business rules and logic.
✓Error handling functions.
✓Navigation, screen management, and report filtering.
✓Security handling, login, user authentication.
✓Data encryption/decryption modules.
✓Handicap accessibility (Americans with Disabilities Act).
✓Data optimization, archival, and restoration utilities.
✓Privacy and data accessibility modules (e.g., HIPAA).
✓Management Reporting.
✓Audit functions.
✓System monitoring and management utilities.
✓System failover and recovery utilities.
✓Data extraction, cleansing, import, and validation functions.
✓System test scripts, drivers, and simulators.
✓On-line help functions.
✓Interactive training materials.

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

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

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

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

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

4.4 CONDUCT INTEGRATION AND SYSTEM TESTING

Purpose

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

Description

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

Roles

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

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

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

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

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

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

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

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

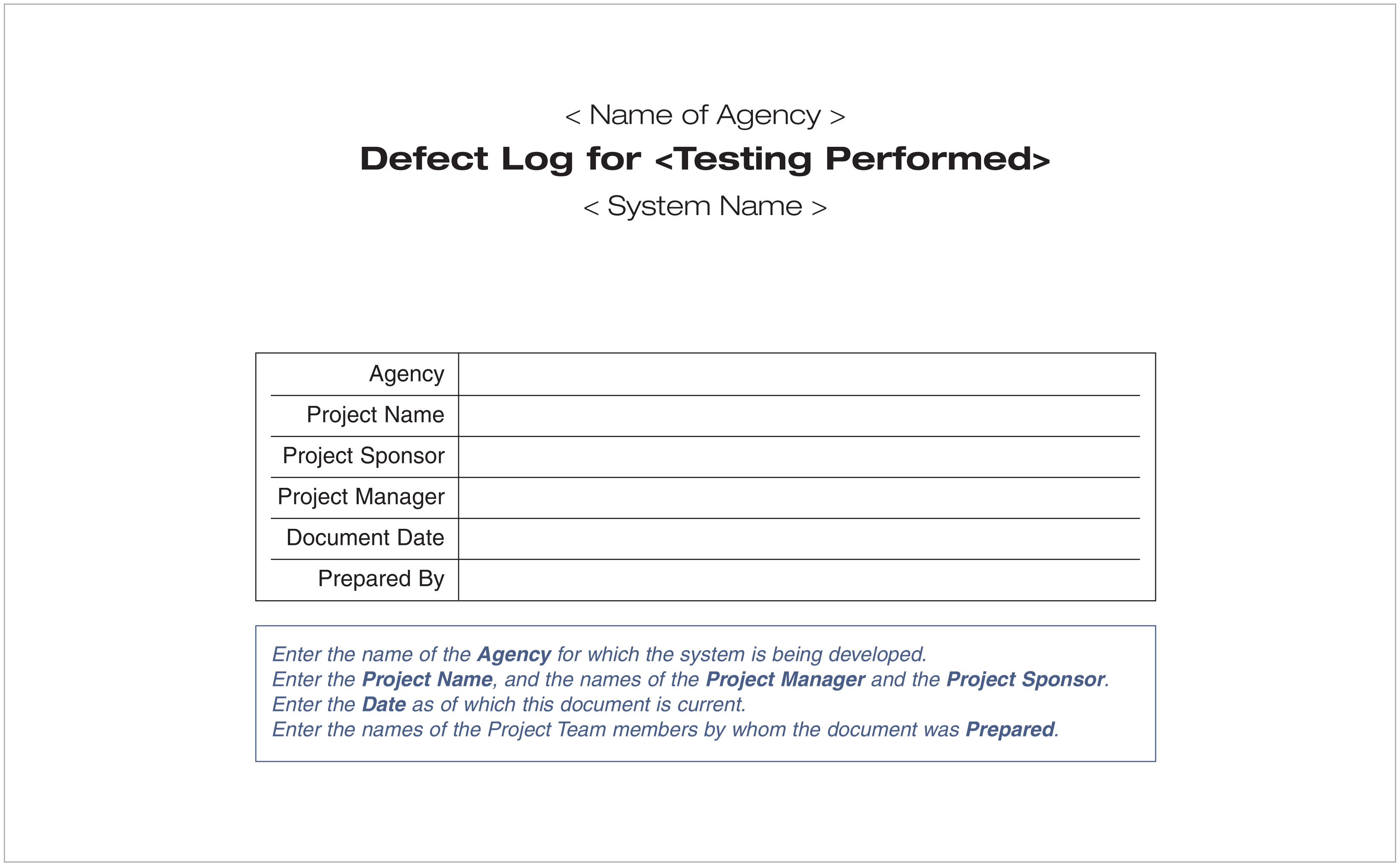
Having completed Integration Testing, System Testing now serves two purposes. First, it is the ultimate Integration Test, incorporating all modules into a single system. Second, it validates the operation of the system as it performs against the anticipated boundary, volume, and peak load conditions.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

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

Deliverable

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

Figure 4-4 Defect Log



< Name of Agency >

Defect Log for <Testing Performed>

< System Name >

Agency	
Project Name	
Project Sponsor	
Project Manager	
Document Date	
Prepared By	

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

Figure 4-4 (continued)

Defect Log for <Testing Performed>

Testing Performed: ____________________________

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

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

4.5 PRODUCE USER AND TRAINING MATERIALS

Purpose

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

Description

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

Roles

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

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

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

Regardless of whether or not the development of training materials requires actual software development (as would be the case should the system include automated or computer-aided training modules), the production of user and training materials still follows the same Build, Test, and Validate cycles as the rest of the system. As each component is created, it should be evaluated against predefined criteria to ensure that it delivers the instructions and concepts it is intended to deliver. Similarly, appropriate quality assurance reviews must be performed to validate that correct standards and conventions are being followed.

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

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

4.6 PRODUCE TECHNICAL DOCUMENTATION

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

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

Roles

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

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

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

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

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

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

Figure 4-5

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

Phase Risks / Ways to Avoid Pitfalls

PITFALL #1 – THROWING THE QA OUT WITH THE SCHEDULE

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

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

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

PITFALL #2 – HIDING IN THE "BLACK BOX"

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

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

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

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

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

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

PITFALL #4 – WHO NEEDS REAL DATA?

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

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

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

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

PITFALL #5 – THE 90% COMPLETE SYNDROME

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

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

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

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

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

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

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

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

? Frequently Asked Questions

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

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

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

What's a test script?

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

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

Where in the lifecycle do you do a stress test?

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