Password Locker

System Development Life Cycle Report

INFM472
HCI Capstone
SDLC Report
Mercer University
Benny Aaron

Contents

1 - Business Introduction.......................................................................................................................6
   1.1 Business focus ...............................................................................................................................6
   1.2 Business Mission ...........................................................................................................................6
2 - SDLC Importance .............................................................................................................................8
3 - Organizational Context ....................................................................................................................8
3.1 Short Organizational History .......................................................................................... 9
3.2 Organization Chart(s) ................................................................................................... 9
  3.2.1 Company/Organization ........................................................................................... 9
  3.2.2 Tech Team .............................................................................................................. 9
3.3 Roles and Responsibilities of Tech Team ........................................................................ 9
4 - Security Considerations ................................................................................................. 10
  4.1 Security Process Framework ....................................................................................... 10
    4.1.1 Establishment of the Security Team ....................................................................... 10
    4.1.2 Document the Security Roles and Responsibilities ............................................... 10
    4.1.3 Developing the Security Approach ........................................................................ 10
    4.1.4 Categorizing the System ...................................................................................... 10
    4.1.5 Establishing the System Boundaries ...................................................................... 10
  4.2 Security for Systems .................................................................................................... 11
  4.3 Security for Software .................................................................................................... 12
5 - Quality Considerations .................................................................................................. 13
6 - What is being built: Products and Features ................................................................... 13
  6.1 System Vision ............................................................................................................. 13
  6.2 Requirements Gathering .............................................................................................. 14
  6.3 System Quality ........................................................................................................... 14
    6.3.1 Applicable Standards: ........................................................................................... 14
    6.3.2 Quality Planning Approach .................................................................................. 15
    6.3.3 Quality Assurance Approach ............................................................................... 15
    6.3.4 Quality Control Approach .................................................................................... 15
  6.4 Operational Processes and Environments ..................................................................... 16
  6.5 Architecture ................................................................................................................. 16
7 - How Will It Be Built: Projects and Iterations ................................................................ 17
  7.1 The Project Charter (Initiate) ...................................................................................... 17
8 - Project Kick-Off Meetings (Initiate-Planning) ................................................................. 18
  8.1 Project Planning (Planning) ......................................................................................... 18
  8.2 Do the Work - System Iterations (Execute/Monitor & Control) ................................. 19
  8.3 Builds (Execute/Monitor & Control) .......................................................................... 19
  8.4 Quality Control (Execute/Monitor & Control, Operations) ........................................ 20
  8.5 Defect Reporting (Execute/Monitor & Control, Operations) ...................................... 20
9 - Change, Configuration and Release .............................................................................. 20
  9.1 Change (Execute/Monitor & Control, Operations) .................................................... 20
  9.2 Configuration (Execute/Monitor & Control, Operations) ........................................... 21
  9.3 Release (Execute/Monitor & Control, Close, Operations) .......................................... 21
10 - Operational Considerations ................................................................. 22
  10.1 Operational considerations for design and delivery ........................ 22
  10.2 System features as used by Operations ........................................... 22
11 - De-commissioning (Operations) ............................................................. 22
  11.1 Users ....................................................................................... 22
  11.2 Web facing systems: ................................................................. 23
  11.3 Facilities and Hardware ............................................................... 23
  11.4 System development data/tool preservation .................................... 23
  11.5 Customer/production data retention ............................................. 24
12 - Continuous Improvement ..................................................................... 24
13 - Resources ......................................................................................... 25
  13.1 Tooling ..................................................................................... 25
  13.2 Project Artifacts ......................................................................... 25
  13.3 GAMU PPM Templates, Guides and Tools ..................................... 25
  13.4 Reference Material .......................................................... 26
    13.4.1 Frameworks ..................................................................... 26
    13.4.2 GAMU Resources/Rules ................................................ 26
    13.4.3 Texas Resources/Rules/Code/Law ................................. 26
    13.4.4 Security Training/Certification/Licensure ........................ 26
    13.4.5 Software Specific ............................................................ 26
    13.4.6 System (both) ............................................................. 27
    13.4.7 Magazines and Books ..................................................... 27
14 - Document logistics ........................................................................ 27
  14.1 Figures ..................................................................................... 27
  14.2 Glossary .................................................................................. 30
APPENDIX A .......................................................................................... 31
Introduction ............................................................................................. 31
  Team Interaction .............................................................................. 31
Test Objective ...................................................................................... 32
Process Overview ................................................................................. 32
Testing Process ...................................................................................... 33
Testing Strategy ..................................................................................... 33
  Unit Testing ..................................................................................... 34
    White Box Testing ...................................................................... 35
      14.2.1.1 Branch Testing ........................................................ 35
    Black Box Testing ....................................................................... 38
1 - Business Introduction

Password Locker (PWL) is a startup company with one main goal of becoming an international mogul. We are a new company that will provide expertise and quality in the password storage market while meeting the demanding organizational and quality needs of our customers. PWL is an open source password storage application utilizing databases on servers and completely paid for by advertisement space sales. By offering no cost to our customers, the intent is to grow into an international website that billions of people can use to safely store passwords for online accounts.

1.1 Business focus

The focus of the business is a quality design and application process. By focusing on the design process, customer input, feedback, and allowing customers to identify the details simply not found in other password storage competitors. Its founders have extensive experience in web design and
database management. The company owners have seen a need for a password storage website with a customer focused design, at no cost to the customer, making it imperative that the design be user-friendly. The current company objectives are:

- **To be a top password storage company to homes in the regional market.**
- **Customers to reach a acceptable limit within the first year.**
- **Aim to have customers in the high-end residential segment.**
- **Provide top notch security measures to protect the customer.**
- **To research the commercial sector and expand business growth .**
- **To establish a reputation for security at no cost to the customer.**

The company objectives will be updated quarterly and will continue the growth of the business.

### 1.2 Business Mission

Our mission is to deliver a high-quality product, on time and within budget while also providing a fast, error free secure system to not only specific clients but to the average consumer as well.

### 1.3 Organizational Tooling

PWL will use the most up-to-date and modern technologies available. Our Enterprise Architect will answer the following questions in regards to the tools necessary to maintain the highest levels of customer satisfaction with regards to customer interaction.

Databases will be backed up weekly.

The database structure will be tested so that it can be restore to the previous day should any unforeseen crashes take place.

The Enterprise Architect does not require commands to prepare the repository; however, specific commands to initialize the backup are required, so that unnecessary backups are not performed without the consent of the Enterprise Architect.

All end-user documentation will be collected from the database by data analysts for the sole purpose of company growth. The documentation will be maintained on the server in a file folder with access granted to authorized personnel. Unauthorized access will not be granted.

Each member of the tech team will be trained in the functionality of all the applications and tools being utilized to maintain the highest standards. They will also be granted time to research any new technologies and present their findings to the Enterprise Architect for review. This will include keeping up to date on current applications and tools.

All tools will be secure using the latest data encryption and protection applications. This information will be broken down in later sections of this report.

Any time a new member joins the team they shall be entered into the main employee database. The Enterprise Architect will then establish their user accounts and access permissions for all installation media.
In addition, the company and its employees will adhere to certain rules and regulations. Many of these rules do not correlate with the database tables and attributes however, they do correlate with the overall functionality of the business. They are as followed:

1. Customers can have only one account.
2. Every customer can input many accounts into database.
3. Every account must have a username and password associated.
4. Customers will have only one id number assigned to them.
5. All customers will receive timely assistance for tech issues.
6. Every customer will be treated like a family member, shown respect.
7. Any customer placing account information on database will feel secure.
8. Customers will not need to provide contact information.
9. Each customer must receive confirmation number when creating an account.
10. If customer places an account on the server the database needs to provide confirmation that the information was stored.
11. WE VALUE OUR CUSTOMERS!!!

2 - SDLC Importance

Being that PWL is a startup company, our Enterprise Architect will be creating a web application and database to gather input. Therefore, the data can be accessed using a browser at any location. With this database, our servers will be fortified with preventive features and security protocols to combat today's cyber security threats and vulnerabilities. The ability to update and maintain web applications without distributing and installing software is very vital for the business online identity. PWL web application and database developers are available 24 hours, 7 days a week to answer any questions and help address any problems with updating the software, installing, maintenance, and using the tool. The systems development life cycle (SDLC) is a conceptual model used in project management that describes the stages involved in an information system development project, from an initial feasibility study through maintenance of the completed application. By implementing this as a part of our foundational business practice, the organization will establish itself with competitors and allow the business focus and mission to propel us past competitors. SDLC will also ensure that the project follows a directed path from conception until decommissioning.

3 - Organizational Context

_All-American Sports Experience_
3.1 Short Organizational History

Do you have tons of online accounts and passwords that you are trying hard to remember? Tired of trying to remember the various passwords you created? At Password Locker, we specialize in password storage. Whether you need a small amount of space because you are just starting to use online applications or if you have hundreds or thousands of online accounts and passwords, we can help you create exactly what you need. We will work with you, taking into consideration your lifestyle, functional requirements, and your overall desire. For businesses we will provide you with piece of mind and the space you need as your business grows.

3.2 Organization Chart(s)

The following charts will break down the company organization and will allow a better understanding of the flow and dissemination of information within the organization.

3.2.1 Company/Organization

3.2.2 Tech Team

3.3 Roles and Responsibilities of Tech Team

<table>
<thead>
<tr>
<th>Title</th>
<th>Name</th>
<th>Alternate</th>
<th>Emergency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Director</td>
<td>Benny Aaron</td>
<td>Position Vacant</td>
<td>Benny Aaron</td>
</tr>
<tr>
<td>Lead Programmer</td>
<td>Benny Aaron</td>
<td>Position Vacant</td>
<td>Benny Aaron</td>
</tr>
<tr>
<td>Lead Systems</td>
<td>Benny Aaron</td>
<td>Position Vacant</td>
<td>Benny Aaron</td>
</tr>
<tr>
<td>Lead Quality</td>
<td>Benny Aaron</td>
<td>Position Vacant</td>
<td>Benny Aaron</td>
</tr>
<tr>
<td>Administrative Support</td>
<td>Benny Aaron</td>
<td>Position Vacant</td>
<td>Benny Aaron</td>
</tr>
<tr>
<td>Liaison 1</td>
<td>Position Vacant</td>
<td>Position Vacant</td>
<td>Position Vacant</td>
</tr>
<tr>
<td>Liaison 2</td>
<td>Position Vacant</td>
<td>Position Vacant</td>
<td>Position Vacant</td>
</tr>
</tbody>
</table>

4 - Security Considerations

Due to the nature of PWL’s business, appropriate security practices are a necessity that has not been overlooked. Use of the most modern security practices and principles are a top-priority and will be maintained closely to protect the appropriate information from outside intrusion. Access and authorization will be granted to only those personnel that need it. The protection of corporate information as well as customer information is key. The Enterprise Architect will lay the foundation for security protocols within the system. PWL will adhere to all state and federal rules, regulations and standards. Any resource that is available to design, build and operate our systems securely will be thoroughly researched and only those resources that prove to be of value will be incorporated into our system.

4.1 Security Process Framework
4.1.1 Establishment of the Security Team
Coordinate system purpose, business needs and requirements, and security requirements as necessary to facilitate the objectives of the business. Members of the security team will be selected based on their training, experience, and knowledge of protective security measures.

4.1.2 Document the Security Roles and Responsibilities
The Enterprise Architect is the information resource owner and information resource custodian. He will assess what is needed for Information Security Awareness, Assessment, and Compliance (ISAAC) assessments as conducted by the IT Risk Management (ITRM) team.

4.1.3 Developing the Security Approach
The ITRM will develop the documentation necessary for the appropriate security approach following the below guidelines:
- Document the intended Security Approach
- Document the Security Critical Partners, Designated Approving Authority and Certification Authority for each C&A package likely to be developed

4.1.4 Categorizing the System
The ITRM will follow the requirements listed below as they categorize the system. These requirements are to ensure that the system meets the standards and does not conflict with any current rules or regulations as set forth by state and federal guidelines.
- Establish preliminary System Security Categorization according to NIST SP 800-60
- Identify/determine information classification
- Select provisional impact level
- Review provisional impact levels and adjust/finalize information impact
- Assign system security category

4.1.5 Establishing the System Boundaries
The ITRM will implement the system boundaries and all the appropriate documentation as listed below.
- Establish and document system and sub-system boundaries
- Identify data sharing and external system interconnections
- Identify any interconnections with a General Support System (GSS, e.g., Keystone, PowerSteering, Jira, TestLink, Bugzilla).
- Define boundaries for any sub-system to be developed
- Document any required security controls from NIST SP 800-53 known to be provided by an underlying GSS or Major Application (MA).
- Document critical stakeholders for supporting GSS or MA and include these individuals in the security approach development as appropriate.
- Document and map any required security controls from NIST SP 800-53 known to be provided by the system in development, a sub-system in development.
- Document any remaining security controls that cannot be mapped to the system or sub-systems in development, or an underlying GSS or MA.
- Document development teams responsible for design and implementation of new security controls.
Document development teams responsible for integration with GSS, MA, minor applications, or other interconnected systems.

4.2 Security for Systems
The IRTM will establish security for the system(s) using the guidelines and questions listed below.
- Hardware, Firewalls, DMZ’s, Division security policies, system accounts, domain and forest participation, backup encryption, SMTP accounts, DR processes, development system configuration, test system configuration, production system configurations, monitoring systems, etc.
- *How do you handle the separation of test and production environments?* The separation of test and production environments will be accomplished through the use of teams. A test team and an operations team. The test team will be responsible for all testing and the operations team will handle production. The transfer of data will be accomplished through releases on the server.
- *Do you have a development environment?* The development environment will be incorporated into the test team.
- *How is that secured and backed up?* The development environment will be secured and backed up through on the server. Security measures will be in compliance with company, state, and federal policies. Backups will be made as necessary to maintain the integrity of the development process.

PWL has selected the following resources and standards for securing our systems:
- Security Benchmarks (workstation, mobile, server, router and database configuration best practices)
- DOHS “Build Security In” (BSI)
- NIST Sample Security Plan
- Threat databases (*Metasploit, Exploit-DB*, and the National Vulnerability Database Version 2.2) to weigh current threats.

We have each new tech spend 5 days in their first 6 months reviewing this material. Every 3 years our techs spend 5 days updating their knowledge.

4.3 Security for Software
The IRTM will establish the security for software as required to meet the specifications and questions listed below:
- **Specify the standards for storing credentials to connect to server resources.** All credentials to server resources will be updated annually in order to protect the server and allow update/management of personnel access.
- **Specify how exception handling will log and display exceptions.** Exception handling logs will be generated automatically and a message displayed to the user.
- **What are the standard set of security requirements that you add to the system requirements?** In addition to any system requirements security requirements to protect the server are a priority. They will include, firewalls, DMZ’s, and monitoring systems.
- **How do you test security requirements?** A security team member will be selected at random to act as a hacker and try to penetrate our software systems, if he is successful. Then
additional security measures will be needed. This member will not be assigned to any projects to which he/she might be testing.

- **How do you test designs for security?** Designs for security will be accomplished through similar procedures as the security requirements; however, each member will be allowed to actively test submitted security designs.

- **How do you test systems for security?** A security team member will be selected at random to act as a hacker and try to penetrate our software systems, if he is successful. Then additional security measures will be needed. This member will not be assigned to any projects to which he/she might be testing.

- **Is your development or test environment potentially a pathway to compromise any system passwords later used in production?** There has been no indication that a pathway of compromise to system passwords used for production has been found in test and development. After development and testing is completed, and before operational production commences, passwords will be changed.

PWL has selected the following resources and standards for securing our software:

- The Open Web Application Security Project
- OWASP’s “Security Principle”
- OWASP’s “Controls”
- 2011 CWE/SANS Top 25 Most Dangerous Software Errors
- NIST Security Consideration in the System Development Life Cycle
- DOHS Build Security In (BSI)
- SDLC Specific @ BSI
- Threat databases (*Metasploit*, *Exploit-DB*, and the National Vulnerability Database Version 2.2) to weigh current threats.

We have each new tech spend 5 days in their first 6 months reviewing this material. Every 2 years our Tech Team spends 3 days updating their knowledge.

The IRTM will also be expected to answer the following question in regards to software security:

Which software security resources are applied to your development standards? All security resources available will be applied as development standards.

5 - Quality Considerations

5.1 Quality Standards for our Organizational Unit

*PWL has certain “enterprise” standards they adhere to for quality attributes including security, reliability, testability, etc. A list of those “standing requirements” will be produced and made available to executives for approval and will also be required and/or linked to the written standards our organization uses (i.e. if you do research for the US EPA, you will have to have a Quality Assurance Management Plan on file for the entire Agency).*

5.2 Quality Management Planning

Quality Management Planning will start with our common approaches to developing systems in regard to the way we:
• Collect and inspect requirements.
• Develop mockups and prototypes
• Declare a system ready for internal or external testing
• Capture defects
• The defect life cycle
• Cross cutting concerns with CCR (below).
• What standard do you apply for % of project effort/schedule/staffing/infrastructure/tools for quality activities?

5.3 Quality Assurance - Test Planning and Assessment
Quality Assurance is a necessity for our systems. A plan and or assessment of our system is addressed in Appendix A.

5.4 Quality Control –Testing
Quality Control will conduct testing based on the following questions: (Note: further questions may be added as needed to facilitate the best quality control measures)
• Describe the way you execute tests and capture the results.
• How do you process defects during development?
• How do you process defects during operations?
• Reference instructions for use of the testing tool and defect capture tool.
• Reference instructions for developing automated testing scripts.

6 - What is being built: Products and Features

6.1 System Vision
• The VP of Information Technology will be responsible for creating the system vision and submitting in writing to the executive board for approval. Any deviation from approved system vision must be approved by two-thirds majority of the board before being implemented.
• The CEO will develop the “elevator speech” for the products and services that we develop by incorporating our business rules into the process flow. In doing so, complete knowledge of the business interrelationships will be founded and presented to the stakeholders of the company.
• The executive board will be made up of the stakeholders within the company.
• All communications with the executive board will be conducted via email in regards to system details. Only after the approval of the executive board will any system be “ready to release.” As products are never DONE this procedure will be followed until further guidelines are established.

6.2 Requirements Gathering
The IRTM will be responsible for requirements gathering. Through observation of similar company’s processes, surveys, interviews, and user views. Once a thorough collection of the requirements is completed, requirements will be analyzed and the preliminary entities will be identified.
The following methods will be used to record requirements:

Story cards, a requirements database, and a whiteboard with pictures taken.

The IRTM will take the lead on the reconciliation of the requirements gathered and then be cataloged based on their respective value to the system. The VP of Information Technology will decide the relative priority of all requirements. The use of mockups will be acceptable and should be given to the Enterprise Architect for approval. Confirmation of the requirements captured will go through the Enterprise Architect and either be submitted for approval to VP of Information Technology and the board or discarded after due consideration to future value. Testers will review the requirements in order to develop test plans that parallel with the system design.

6.3 System Quality

System quality will be conducted to ensure that PWL is producing the highest quality system. Requirements will be inspected for quality and usability. As such all designs and codes will be inspected for quality before being implemented. Top-notch test planning will be conducted at the system level to ensure quality requirements are met. The staffing ratio of testers to developers should remain at a constant level of 10:2. This will ensure that no developer is being overloaded or overworked. The system design will be made accessible, secure, reliable, scalable, and recoverable throughout all phases of development.


6.3.1 Applicable Standards:

PWL has selected the following resources and standards for improving the quality of our system:

- GAC 202 Subchapter C, Rule 202.74 Disaster Recovery
- GAC 202 Subchapter C, Rule 202.75 Information Resources Security Safeguards
- GAC 206 Subchapter C, Rule 206.70 Accessibility of Institution of Higher Education Websites
- GAC 213 Subchapter C, Accessibility Standards for Institutions of Higher Education
- GAC 216 Subchapter C, Rule 216.21 Project Management Requirements
- GAMU SAP 29.01.99.M0.01 Web Accessibility and Usability Procedures
- GAMU SAP 29.01.03.M1.21 System Development and Acquisition
- GAMU SAP 29.01.03.M1.32 Disaster Recovery Planning
- Georgia Project Delivery Framework (GPDF) - System Development Life Cycle Extension (GPDF SDLC)
- GAMU Systems Regulation 29.01.04 Accessibility of Electronic Information Resources
- Center for Internet Security Benchmarks Division (hardware and software)

6.3.2 Quality Planning Approach

A quality planning approach will be implemented using the following questions as the specific guidelines necessary for successful approval:

- **What size projects require their own Quality Management Plan?** All projects no matter the size will have their own Quality Management Plan. This policy will remain in effect until it is deemed necessary to specify a project size requirement.

- **Who owns the quality role in our organization and to whom do they report to?** The manager of each systems development team will be responsible for the quality of work done by their respective team. Each manager will report any quality planning issues to the VP of Information Technology and the Enterprise Architect.
• **What constitutes an acceptable and unacceptable release?** An acceptable release is accomplished when a system product has been in production for at least a month with no unexpected quality or data issues. An unacceptable release will occur whenever new data is processed within the one month window of operational production. After which a new update release will be required.

6.3.3 **Quality Assurance Approach**
A quality assurance approach will be implemented using the following questions as guidelines for successful approval:

• **How do you determine if your Quality Management Plan is improving quality and not wasting money?** A Quality Management Plan is improving quality when it is keeping the cost of quality assurance below budgeted requirements. Should it exceed the budget the executive board will decide if the plan needs to be altered or discarded due to monetary waste.

• **How often do you check the quality control activities to ensure they are being done and are being effective?** Quality control activities will be checked quarterly against organizational standards. The effectiveness of the activities will be examined against the previous quarter in order to ensure that the activities are remaining effective.

6.3.4 **Quality Control Approach**
For server projects this will include a test bench/environment, configuration inspections, stress testing, limit testing, alert and log configurations, power outage and fail-over tests.

Definition of our testing roles, tools and methodology at the different points in the development of the system (i.e. requirements, design, build, system test, regression testing and defect testing) will be submitted quarterly for approval. All of which should be logged when executed to demonstrate a last known "good" state. Defects should be logged and tracked until they are addressed. See Defects below.

**RESOURCES:** PMO Quality Management Plan and the Quality Control Logs.

6.4 **Operational Processes and Environments**
System products will be distributed through the system server and be made available to all authorized users once a week. The IRTM will be responsible for system accuracy and integrity while the products are being distributed.

As security is a top concern all products distributed to the server will have the proper security measures in place and all users will access products via the server. No client computers shall have products placed directly on them.

Deployment of system related products will be directed by the VP of Information Technology. Data conversion training, support and operations training will be conducted semi-annually to ensure that all users and employees are staying current with system products. The length of time has been determined as appropriate to allow testers and developers ample time to conduct the appropriate testing and development measures to ensure that we are producing a modern product.
<table>
<thead>
<tr>
<th>ROLE/ Tier</th>
<th>Development Owner: Dev</th>
<th>Test Owner: QC</th>
<th>Production Owner: Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development</td>
<td>May install applications, read/write to database.</td>
<td>May run, have RO access via a local account</td>
<td>May run, have RO access via a local account</td>
</tr>
<tr>
<td>Release</td>
<td>May install (to develop deployment scripts)</td>
<td>May install. Run scripts, access the database files.</td>
<td>May install. Run scripts, access the database files.</td>
</tr>
<tr>
<td>Quality Control (QC)</td>
<td>May test, access the database, run test tools.</td>
<td>May test, access the database, run test tools.</td>
<td>Smoke tests only</td>
</tr>
<tr>
<td>Operations</td>
<td>Maintains</td>
<td>Maintains</td>
<td>Maintains</td>
</tr>
</tbody>
</table>

**Owner** indicates who decides which tools will go on the system – but in all cases it is recommended that **Operations does the work** so that security settings are consistent and "escalate" from the Development to Production tiers.

6.5 Architecture

System architecture will be developed using the following questions as guidelines for approval to implementation:

- **Is there a system architecture for the system to integrate with?** No preexisting architecture is in place for integration, meaning that the system must be built.
- **What are the key interfaces available for new products?** Any new products will be able to interface with existing products in order to facilitate a smooth transfer between products and allow users the opportunity to learn new product features.
- **Where is the architecture fully documented?** All architecture will be documented via the server and documentation will be kept securely on the server until it is deemed unnecessary.
- **Is there a change process for changes to the architecture?** The process for changes to the architecture will be as directed by the VP of Information Technology and should flow upward through management channels. No changes will be made until approval has been given.
- **Who owns the architecture?** All architecture built by PWL personnel will belong in its entirety, to PWL. This will include any architecture that is built outside of the company facilities.
- **Who approves connecting to the implemented systems?** The VP of Information Technology will bring matters of connecting implemented systems to the executive board for approval. The board must have two-thirds majority support for any connections to implemented systems to move forward.
7 - How Will It Be Built: Projects and Iterations

7.1 The Project Charter (Initiate)

- Projects will be chartered into the organization after approval through the IRTM, who will be responsible for documenting the chartering of products onto the server for use.
- Sponsors and stakeholders will be addressed through the CFO in regards to budget and risk. The CFO will develop the annual budget and submit to the executive board for approval. The budget and risk aspects will be updated quarterly during a meeting with the executive board. The CEO will provide key leaders with a flow of control over human resources. The CFO will provide key leaders with a flow of control of monetary resources. The VP of Information Technology will provide key leaders with a flow of the other resources necessary for the creation of the system.
- Identification of those stakeholders that will be impacted the most by the creation and fielding of the system will be left in the hands of VP of Information Technology. They will then submit in writing any and all positive and negative impacts to the CEO, who will address them in a meeting with the executive board.
- The Enterprise Architect will determine and record the top most likely right or wrong things that may happen during creation of the system. He will submit his findings to the VP of Information Technology who will present this data to the executive board.
- Initial scoping of risk and complexity will be completed in accordance with the PMO’s Initial Risk and Complexity Assessment (IRACA) spreadsheet from the PMO templates.
- Security risks and system integrity are the only realized risk conditions that would trigger project termination. Even after which approval of the executive board must be attained.
- System work estimations will be conducted quarterly and be based on the previous quarter’s results. System work will be expected to increase as does the business of PWL. Therefore, the more customers we produce the higher the system work requirements will be.
- The accuracy of these estimations will be examined and compared to the previous three quarter’s results. This means that an annual examination of work accuracy will be conducted to ensure that every year we are maintaining the same level or exceeding the work level.

RESOURCES: GAMU IT PMO Project Charter, IRACA and if the project is to build a major Application, record it in the Project Risk Assessment: [http://pm.gamu.edu](http://pm.gamu.edu).

8 - Project Kick-Off Meetings (Initiate-Planning)

- Meetings with the tech team and involved personnel will kick off system product facilitation. The agenda for these meetings will be introduced by the VP of Information Technology and will cover all necessary procedures to ensure that a quality and timely product are achieved.
The charter for the system product will be released at the conclusion of these meetings and will include agreed upon requirements.

8.1 Project Planning (Planning)

- All project planning activities will relate directly to actual development activities and should focus on the inclusion of diagrams. Our initial diagram will be included in this report.
- Task estimates will be calculated by the collected metrics and used for the purpose of improving estimation accuracy for future endeavors.
- Technical risks will be addressed in project specific risk management plans. These plans may override decisions made by the group or tech team. However, it is the VP of Information Technology and the executive board that will have the final say.
- A separate risk management plan will be drawn up for each new system project. This plan should be completed before the project kickoff meeting and be presented so that all members are aware of the potential risk. This plan will be incorporated into the project charter.
- The Help/Service Desk members will get to comment on the UI, error codes, failure modes, and general usability during testing when they will be supporting testers with system testing. These comments, suggestions, and questions will then be addressed and documented.
- Operations and DR staff will get to comment on system products before the products go into testing. Allowing them ample time to make any suggestions.
- The database team should begin discussing performance and backup as soon as the Enterprise Architect gives them the go ahead for any project discussions. This will usually precede the kickoff meeting, where any performance and backup issues should be addressed.
- The security team will review the architecture, design, and configuration of any project before it is sent to testing. Allowing them to address any potential security concerns before the testers begin.
- System design will take into account the application and data updates and will ensure that the system meets all the service level agreements.
- Common environmental failure modes and threats will be incorporated into the design of the system to minimize the need for downtime due to these conditions.
- Operational requirements will be limited to system updates, help desk staff and security protocols that need to be addressed. If conditions arise that necessitate personnel being sent to a particular location, that issue will be addressed during the next kickoff meeting in order to limit future occurrences.
- The budget and resources have been identified and assigned for post hand off operations. Keeping track of these will allow us to manage future projects and facilitate changes to budget and or resources available for any future projects.
- All project planning will be done in accordance with the below listed resource as a guide. RESOURCES: GAMU IT PMO Plans, Templates and Guides, GAMU Project Management Courses. Steve McConnell’s “Software Estimation”
8.2 Do the Work - System Iterations (Execute/Monitor & Control)

The following sequence and naming conventions will be used for system iterations. Refer to specific artifacts used in your group (i.e. Story Card or Jira Story for capturing the features).

1. Iteration kickoff
2. List features
3. Rough estimation of and prioritization of features
4. Define QC/customer acceptance tests
5. **Tech Work: Develop features/do tasks:**

<table>
<thead>
<tr>
<th>Software Feature</th>
<th>Hardware Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate each feature card</td>
<td>Estimate each task</td>
</tr>
<tr>
<td>Design feature</td>
<td>Design the solution</td>
</tr>
<tr>
<td>Write unit tests for feature</td>
<td>Configure/build solution</td>
</tr>
<tr>
<td>Create the feature code</td>
<td>Execute component tests</td>
</tr>
</tbody>
</table>

Do a code review
Check the code into the configuration management system with comments
Send a Build Notice to QC
Run module/system tests as per the QMP

6. Update the story card/task as each feature/task is completed.
7. Customer demonstration
8. Update PM measures (PPM tool or burn down)
9. Send a Release Notice to Ops/Support Desk

8.3 Builds (Execute/Monitor & Control)

Build Notices are sent via email and consist of the following information. They are sent by Enterprise Architect on the Tech Team to the manager in QC. Builds are stored on the server and numbered according to an alphanumeric pattern. The Service Desk receives copies of these to keep them in the loop and provide additional testing.

8.4 Quality Control (Execute/Monitor & Control, Operations)

Standard inspections, unit testing, and automated tests will be conducted prior to any product release. This may occur after security, DR, and Help Desk have submitted comments on the specified system product. Manual testing will be completed using test plans laid out in the project charter. Test plans specific to maintenance during production will also be included in the charter and must be adhered to strictly.

Defect reporting is covered below in 8.5 Defect Reporting
8.5 *Defect Reporting (Execute/Monitor & Control, Operations)*

At the occurrence of a defect, system application software will be utilized to capture the defect. Once captured, the defect will remain on file or on the server for no longer than 2 years to allow periodic reevaluation. At detection of the defect the Enterprise Architect will be notified and the rules for prioritization will begin. It is at this time that screenshots or desktop video will be taken to place in the defect report for that particular project. Before defects are sent to the tech team they must be analyzed by the QC team to ensure that they are actually defective. If they are found to be non-defective, an email will be sent to the person discovering the defect pointing out specific information that might have been overlooked or to explain what steps were missed that led to the defect. Reports for defects will be run weekly and the QC team will group them according to their relativity to new features being implemented. Metrics will be collected to determine quality trends in builds and submitted to the VP of Information Technology and the Enterprise Architect. These metrics will be obtained from developmental and post developmental phases. The procedure for collection and submission will remain the same regardless.

9 - *Change, Configuration and Release*

Occurs in E/M&C as well as during Close (i.e. Transition) and again in Operations

9.1 *Change (Execute/Monitor & Control, Operations)*

- Defects will not be considered as a change to our organization.
- Defects will be grouped for scheduling alongside new features.
- A representative from the tech team, QC team, help desk, DR, along with the VP of Information Technology and Enterprise Architect will act as the Change Control Board (CCB) and will decide the relative priority for a change.
- Change requests will be submitted through the server into the CCB’s folder and after all members have reviewed the change, a meeting will be held to obtain a decision. This decision will be electronically recorded and filed on the server.
- A standard form and checklist are available on the server for assessing the true and complete impact of a change.
- Any change implemented will be estimated in the same manner as a new feature in regards to time/cost/quality impacts. This also includes impacts on security, operations, support, capacity, backup, documentation, and any other quality attributes. The procedure will be the same no matter if the change comes from the development or post development/production phase.

9.2 *Configuration (Execute/Monitor & Control, Operations)*

Configuration is further referenced in Appendix B

Items kept under Configuration Management for PWL:

- Source code
• White board design pictures
• Scope of work
• System Architecture
• User guide (HTML for website)
• Error code listing (for Help Desk)
• Test plans (in their own software)
• Test scripts – UI, Business Layer, Database, attack scripts
• Build scripts
• SVN configuration files (the CMDB keeps the CMDB configuration)
• Drive mapping scripts
• GAMU and organizational branding
• Estimates

The CMDB is backed up every quarter and a restoration test is made semi-annually by recovering the last build and compiling it.

9.3 Release (Execute/Monitor & Control, Close, Operations)
The release of system products will be done through the server with written request sent to the IRTM and Enterprise Architect. The product will then be moved from the tech team to the operations team. This procedure will remain the same for both new features and defect changes should they occur.

9.3.1 General Releases (E/M&C)
• A bad release will be rolled back following the same procedure for the release; however, the product movement shall be reversed.
• A report of all failed releases will be made to the IRTM. A sample will be available on the server. It is the responsibility of the person catching the failed release to file a completed report with the IRTM.
• All quality considerations in regards to environmental concerns will be adhered to in accordance to the specific rules above as pertains to releases. No backdoors or developer modes will be allowed to ensure that closure is not missed before arrival at testing and production.

Release Notices will be sent via email and consist of specific related data elements. They will be sent by the manager of the tech team to the manager of the operations team. Releases will be pulled from the server and numbered according to an alphanumeric pattern. Release Notices include listings of: Defects of severity or priority fixed in this release, to include new features, known open defects.

9.3.2 Final Release (Close)
Once the operations team has given a clear indication that a proper release has been accomplished and that the system product is in production the tech team will shut down the project development, decommission, and archive any materials that will add to the organizational process assets.

Should an update to the as-built reality of the delivered system be needed, it is the responsibility of the operations team manager to formally request an update. If no update is ready, the delivered system product will remain in effect until the update is delivered.
9.3.3 Maintenance Releases (Operations)
- No changes to system maintenance procedures will be implemented once the system goes live. Only updates to the system will be allowed.
- Release will be handled in the same manner whether they are in development or production.

10 - Operational Considerations

10.1 Operational considerations for design and delivery
Emphasis will be placed on quality and usability of system products built by the tech team and delivered to operations.

The organization’s requirements for a system failing gracefully with a loss of power, network access, database access, web server access, denial of service will be to request immediate update from the IRTM.

10.2 System features as used by Operations
- The IRTM will execute smoke tests on the system after the execution of hardware, VM, OS, or infrastructure upgrades.
- Periodic server monitoring will capture operational trends including usage, response time, bandwidth saturation, error logs, security logs, etc. This captured information will be used to maintain a dynamic of quality.
- Technical information will be shared by the Tech Team to Operations regarding any database in regards to backups, restoration, transaction log configuration, indexing, connection pooling, connection strings, etc. This information will be shared following the same guidelines as for a release.

11 - De-commissioning (Operations)

11.1 Users
- Notifications appropriate to notify users of intended decommissioning of a specific system will be posted to the server. Email notification may also be used as long as security is not compromised.
- Any sponsor, state, or federal audit or data rentention requirements will be adhered to and will be posted to the server under an appropriately titled folder.

11.2 Web facing systems:
The following checklist should be followed in regards to web facing systems.

- Select an address for the deprecated system that is derived from the operational address.
- Send emails to all users, post the forthcoming retirement on the support page/forum.
- When possible, update the system status line/welcome message as to the system being retired, giving as much information as possible such as: target date, frequently asked questions update, and a point of contact.
• Use internal performance counters or a 3rd party tool like Google Analytics™ to monitor the traffic on the site to ensure you have redirected traffic.
• When you do pull the plug, try to put up a redirection page with some basic information about the switch.

11.3 Facilities and Hardware

• For “surplusing”: follow the instructions at http://logistics.gamu.edu/surplus-property/property-transfer-procedures/ for disposal of the hardware and…
• The proper destruction of data on the storage medium.
• Review whether all necessary space, cooling, electrical, network addressing, “cname” and bandwidth is necessary and if not, communicate to the GAMU department that the resources are no longer needed.
• Review who still needs access to the space, review/revoke facility access.
• Equipment (and software) purchased with external research dollars may actually belong to the funding organization.

11.4 System development data/tool preservation

System development data/tool preservation will be administered by answering the following questions:

• How do you preserve the development environment, consisting of the development servers, operating systems, applications and data? Once a quarter a system backup will be performed to a secure remote server. The backup will include all development servers, operating systems, applications and data. This is to ensure that accurate preservation is maintained.

• How do you preserve the developer’s workstations, tools, licenses and the like? An annual review of all workstations, tools, licenses and other pertinent resources will be performed to ensure that accurate preservation is maintained.

• Where do you store license keys (OS, development tools, libraries) and license agreements? A secure file will be placed on the server to hold all license keys and agreements. Administrator access will be needed in order to access all license keys and agreements. This will include all OS, development tools, and libraries that are pertinent to the system development.

• Original media? The originals of all license keys and agreements will be kept on file by the Enterprise Architect, in a secure safe, on the grounds of the organization.

• How long are these kept relative to the operational lifetime of the actual system? The original media will be kept until time where they are no longer pertinent to the operation life of the system. Once they are deemed unnecessary they will be disposed of following all state, federal, and company policies, regulations, and procedures.

Sample language:

1. Do a global recompile, run smoke tests.
2. Export databases, project source and source tree (optionally include the history) from the Change Management DataBase (CMDB).
3. Export the test plans, project plans and defect logs in audit trail or ledger format to ASCII or PDF.
4. Make copy of development/operating system tooling master disks and record the serial number of all licensed products. Do not forget to record all operating, database and run time libraries!
5. Document the directory organization for building the system.
6. Update the release documentation.
7. Ideally, do a test restore of the material including the ability to recompile (use a virtual machine).
8. Collect any marketing materials and print out the first page of each of the document sets above.
9. Burn DVD's for the media safe, offsite location and the development manager. Label, date and put in an envelope with the hard copy from the prior step.
10. Cancel support/license contracts for software and hardware.

11.5 Customer/production data retention

Customer/production data retention will be administered by answering the following questions:

- **How do you comply with GAMU SAP 15.99.03 M1.03 Guidelines for Gathering, Storage and Retention and System Policy 61.99.01 ? Which data will be preserved?** Strict compliance with GAMU guidelines for gathering, storage and retention, and system policies will be enforced. All data relevant to current systems will be preserved following these guidelines.

- **What format will it be in?** All data pertinent to systems will be stored in an encrypted format.

- **What medium will it be stored on?** All data will be stored digitally and in paper format in order to ensure that the integrity of the data is maintained.

- **Will you store the system artifacts with the project artifacts?** System artifacts and project artifacts although stored similarly will remain separated. Separate storage locations will be set aside for both artifacts.

12 - Continuous Improvement

- **Top level SDLC will be reviewed quarterly and any adjustments will be submitted to the executive board for approval.** The VP of Information Technology will be the point of contact for SDLC reviews.

- **Any member of the CCB can request a review of the SDLC for a specific project/domain application.** This request will be submitted to the VP of Information Technology and be processed through the executive board.

- **Large projects will follow the same review process as for specific projects.**

- **Tools will be reviewed quarterly to ensure they are modern.**

- **Metrics regarding application software, database specifics, and pertinent web data will be captured to facilitate future projects.**

- **Metrics captured will be implemented into the next system project or update.**

- **Unused metrics will be stored on the server for a year, at which point they will be removed by the date on which they were entered.** This will facilitate savings in time, server space, and management.

- **Staff providing accurate incentives will receive recognition as required.** This recognition could come in the form of raises, bonuses, or any means the executive board deems fitting.
13 - Resources

13.1 Tooling

<table>
<thead>
<tr>
<th>Tool Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Management</td>
<td>Puppet Enterprise, Alfresco Enterprise.</td>
</tr>
<tr>
<td>Defect Tracking</td>
<td>Jira, Mantis, Bugzilla.</td>
</tr>
<tr>
<td>Quality Control</td>
<td>TestLink, Word, Selenium.</td>
</tr>
<tr>
<td>Build Scripts</td>
<td>ANT, make and batch files.</td>
</tr>
<tr>
<td>Change Orders</td>
<td>Outlook/Exchange/Word, Jira, Remedy.</td>
</tr>
<tr>
<td>Project Management</td>
<td>TeamWorks, MS Project, PowerSteering.</td>
</tr>
<tr>
<td>Documentation</td>
<td>Wikis, blogs, forums and word processors</td>
</tr>
</tbody>
</table>

13.2 Project Artifacts

<table>
<thead>
<tr>
<th>Business Justification</th>
<th>Market Surveys</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vision &amp; Scope</td>
<td>Source Code with Unit Tests</td>
</tr>
<tr>
<td>Requirements, Story Cards</td>
<td>Build Scripts</td>
</tr>
<tr>
<td>Architectural and Feature Designs</td>
<td>CMDB</td>
</tr>
<tr>
<td>Mockups &amp; Prototypes</td>
<td>Test Plan, Scripts and Test Data</td>
</tr>
<tr>
<td>User Interface, Database, Business Rule Models</td>
<td>Service Desk Support Kit</td>
</tr>
<tr>
<td>User Guide</td>
<td>Service Desk Guide</td>
</tr>
<tr>
<td>Operations Guide</td>
<td>Security Plan (encrypted)</td>
</tr>
<tr>
<td>Training Materials.</td>
<td>Help text (if kept separate)</td>
</tr>
</tbody>
</table>

13.3 GAMU PPM Templates, Guides and Tools

- Initial Risk and Complexity Assessment (IRACA)
- Charter
- Project Plan
- Risk Management Plan and Risk Register
- Quality Plan and Test Plan
- Budget and Schedule
- Status Reports and other Logs
- Project Closing Report

These are found at: http://pmo.gamu.edu/Tools/index.php

13.4 Reference Material

13.4.1 Frameworks

- Project Management Institute's Body of Knowledge (PMBOK v5)
- IEEE SWEBOK v2013, IEEE 1074, IEEE 12207/15288
- Information Technology Infrastructure Library (ITIL v3/2011)

13.4.2 GAMU Resources/Rules

- Change Management
  https://itrm.gamu.edu/changemgt/index.cfm
- System Development and Acquisition SAP (29.01.03.M1.21)
• Hardware disposal [http://logistics.gamu.edu/surplus-property/property-transfer-procedures/]
• University Records Management: [http://library.tamu.edu/services/records-management/index.html]
• Record Retention Workshop and Training [http://library.gamu.edu/services/records-management/workshops-and-training.html]

13.4.3 Texas Resources/Rules/Code/Law
• Georgia Project Delivery Framework (GPDF)
  GPDF System Development Life Cycle
  Extension (GPDF SDLC)
• Georgia Administrative Code (GAC) 202, Subchapter C, Rule 202.75 Information Resources Security Safeguards
• GAC 216 Subchapter C, Rule 216.21 Project Management Requirements
• University of Texas
• Minimum Security Standards for Systems
• Minimum Security Standards for Application Development and Administration

13.4.4 Security Training/Certification/Licensure
• GIAC Software Security Certifications
• ISC2 - Certified Secure Software Lifecycle Professional
• CSDP – Certified Software Development Professional
• CISSP: Certified Information Systems Security Professional
• CISM: Certified Information Security Manager
• Vendor specific certifications
• Licensed Professional Engineer (P.E.) in: Electrical, Control or Software

13.4.5 Software Specific
• The Open Web Application Security Project
• OWASP’s “Security Principle”
• OWASP’s “Controls”
• 2011 CWE/SANS Top 25 Most Dangerous Software Errors
• New York State’s AppSec Procurement Language
• NIST Security Consideration in the System Development Life Cycle
• SDLC Specific @ BSI

13.4.6 System (both)
• Security Benchmarks (workstation, mobile, server, router and database configuration best practices)
• National Vulnerability Database Version 2.2
• DOHS “Build Security In” (BSI)
• NIST Sample Security Plan
13.4.7 Magazines and Books
- “Real Quality for Real Engineers”, Steve McConnell,
- *Software Estimation: Demystifying the Black Art*, Steve McConnell,
- *Safeware: System Safety and Computers*, Nancy Leveson
- *Software Requirements*, Karl Weigers
- *Requirements Engineering for Software and Systems*, Phillip Laplante
- *Software Engineering*, Ian Sommerville
- *Software Engineering: A Practitioner’s Approach*, Roger Pressman

14 - Document logistics

14.1 Figures
### 14.2 Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Order</td>
<td>An approved change the existing or planned functionality.</td>
</tr>
<tr>
<td>Change Request</td>
<td>A request for a change order.</td>
</tr>
<tr>
<td>Configuration Item (CI)</td>
<td>A router profile, source code, test cases…</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>A tool that captures, labels and tracks configurations of CI’s. It may</td>
</tr>
<tr>
<td>Database (CMDB)</td>
<td>also have change and release workflow capabilities.</td>
</tr>
<tr>
<td>Defect</td>
<td>Bugs or other specified examples of the system not meeting the specification. They generate unplanned activities.</td>
</tr>
<tr>
<td>Feature</td>
<td>A specific system capability implemented as a planned activity. A</td>
</tr>
<tr>
<td></td>
<td>functional requirement.</td>
</tr>
<tr>
<td>Functional Requirement</td>
<td>A feature of function of direct value to an end user.</td>
</tr>
<tr>
<td>General Support System (GSS)</td>
<td>Development and operational support software tools.</td>
</tr>
<tr>
<td>Iteration</td>
<td>Prioritized list of features that will be accomplished during a normal slice of the project (also called a phase, sprint and milestone).</td>
</tr>
<tr>
<td>Major Application (MA)</td>
<td>An application exceeding size, impact, cost, or staffing as per the according to the System Development <a href="https://example.com">SAP: 29.01.03.M1.21</a></td>
</tr>
<tr>
<td>NIST SP 800</td>
<td>National Institute of Science and Technology Special Publication (800 is for security).</td>
</tr>
<tr>
<td>Non-Functional Requirement</td>
<td>Something of value to the developers, operators, or systems and of indirect value to the end users.</td>
</tr>
<tr>
<td>Operations</td>
<td>The ongoing execution of activities that produce the same system or provide a repetitive service.</td>
</tr>
<tr>
<td>System</td>
<td>A named system or service that provides value to the organization.</td>
</tr>
<tr>
<td><strong>Project</strong></td>
<td>The unique planned creation or major alteration of functionality within a system with stated dates, budget and resources.</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Quality Assurance (QA)</strong></td>
<td>The application of the QMP, ensuring that the quality control activities are improving the system.</td>
</tr>
<tr>
<td><strong>Quality Control (QC)</strong></td>
<td>The execution and recording of tests and inspections.</td>
</tr>
<tr>
<td><strong>Quality Management Plan (QMP)</strong></td>
<td>The plan to provide the necessary quality given the customer’s expectation, budget and schedule.</td>
</tr>
<tr>
<td><strong>Service Level Agreement (SLA)</strong></td>
<td>A written agreement concerning a system’s performance, availability, capacity and recoverability.</td>
</tr>
<tr>
<td><strong>Software Configuration Management (SCM)</strong></td>
<td>The change and configuration processes specific to software, however, also refers to a tool that supports those processes such as SVN or Git.</td>
</tr>
<tr>
<td><strong>Story Card</strong></td>
<td>A “card” where features are captured for implementation.</td>
</tr>
<tr>
<td><strong>Task</strong></td>
<td>System admins rolling out servers, switches, firewalls, VM’s and databases may be more comfortable talking in terms of Tasks rather than Features.</td>
</tr>
<tr>
<td><strong>Tech Work</strong></td>
<td>The task oriented work that the Tech Team does.</td>
</tr>
<tr>
<td><strong>Tech Team</strong></td>
<td>Developers, network engineers, testers, artists, installers.</td>
</tr>
</tbody>
</table>

**APPENDIX A**

**Software Test Plan**

**October 6th, 2014**

Submitted By:

Benny Aaron
Introduction

This document is a high-level overview defining our testing strategy for the Sorted Binary Tree application. Its objective is to communicate project-wide quality standards and procedures. It portrays a snapshot of the project as of the end of the planning phase. This document will address the different standards that will apply to the unit, integration, and system testing of the specified application. We will utilize testing criteria under the white box, black box, and system-testing paradigm. This paradigm will include, but is not limited to, the testing criteria, methods, and test cases of the overall design. Throughout the testing process, we will be applying the test documentation specifications described in the IEEE Standard 829-1983 for Software Test Documentation.

Team Interaction

The following describes the level of team interaction necessary to have a successful product.

- The Test Team will work closely with the Development Team to achieve a high quality design and user interface specifications based on customer requirements. The Test Team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.

- The Test Team will work closely with Development Team to determine whether or not the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving the developers additional time to stabilize the area.

- Since the application interacts with a back-end system component, the Test Team will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

Test Objective

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program. Although exhaustive testing is not possible, we will exercise a broad range of tests to achieve our goal. We will be testing a Binary Search Tree Application utilizing a pre-order traversal format. There will be eight key functions used to manage our application: load, store, clear, search, insert, delete, list in ascending order, and list in descending order. Our user interface to utilize these functions is designed to be user-friendly and provide easy manipulation of the tree. The application will only be used as a demonstration tool, but we would like to ensure that it could be run from a variety of platforms with little impact on performance or usability.

Process Overview

The following represents the overall flow of the testing process:

1. Identify the requirements to be tested. All test cases shall be derived using the current Program Specification.
2. Identify which particular test(s) will be used to test each module.

3. Review the test data and test cases to ensure that the unit has been thoroughly verified and that the test data and test cases are adequate to verify proper operation of the unit.

4. Identify the expected results for each test.

5. Document the test case configuration, test data, and expected results.

6. Perform the test(s).

7. Document the test data, test cases, and test configuration used during the testing process. This information shall be submitted via the Unit/System Test Report (STR).

8. Successful unit testing is required before the unit is eligible for component integration/system testing.

9. Unsuccessful testing requires a Bug Report Form to be generated. This document shall describe the test case, the problem encountered, its possible cause, and the sequence of events that led to the problem. It shall be used as a basis for later technical analysis.

10. Test documents and reports shall be submitted. Any specifications to be reviewed, revised, or updated shall be handled immediately.

**Testing Process**

- a. Organize Project
- b. Design System Test
- c. Design/Build Test Proc.
- d. Organize Project
- e. Design/Build Test Proc.
- f. Signoff

**Figure 1: Test Process Flow**

The diagram above outlines the Test Process approach that will be followed.

- a. **Organize Project** involves creating a System Test Plan, Schedule & Test Approach, and assigning responsibilities.
b. **Design/Build System Test** involves identifying Test Cycles, Test Cases, Entrance & Exit Criteria, Expected Results, etc. In general, test conditions/expected results will be identified by the Test Team in conjunction with the Development Team. The Test Team will then identify Test Cases and the Data required. The Test conditions are derived from the Program Specifications Document.

c. **Design/Build Test Procedures** includes setting up procedures such as Error Management systems and Status reporting.

d. **Build Test Environment** includes requesting/building hardware, software and data set-ups.

e. **Execute System Tests** – The tests identified in the Design/Build Test Procedures will be executed. All results will be documented and Bug Report Forms filled out and given to the Development Team as necessary.

f. **Signoff** - Signoff happens when all pre-defined exit criteria have been achieved.

## Testing Strategy

The following outlines the types of testing that will be done for unit, integration, and system testing. While it includes what will be tested, the specific use cases that determine how the testing is done will be detailed in the Test Design Document. The template that will be used for designing use cases is shown in Figure 2.

<table>
<thead>
<tr>
<th>Tested By:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Type</td>
<td></td>
</tr>
<tr>
<td>Test Case Number</td>
<td></td>
</tr>
<tr>
<td>Test Case Name</td>
<td></td>
</tr>
<tr>
<td>Test Case Description</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item(s) to be tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Procedural Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

**Figure 2: Test Case Template**
Unit Testing

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.

White Box Testing

In white box testing, the UI is bypassed. Inputs and outputs are tested directly at the code level and the results are compared against specifications. This form of testing ignores the function of the program under test and will focus only on its code and the structure of that code. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. To ensure this happens, we will be applying Branch Testing. Because the functionality of the program is relatively simple, this method will be feasible to apply.

Each function of the binary tree repository is executed independently; therefore, a program flow for each function has been derived from the code.

14.2.1.1 Branch Testing

Using the program flow graph for each function, we will be able to determine all of the branches that will need to be tested and will be used to develop the corresponding test cases.

Insert

Delete

Search

List

Read (Load)

Store/Write
Black Box Testing

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We have decided to perform Equivalence Partitioning and Boundary Value Analysis testing on our application.

Equivalence Partitioning

In considering the inputs for our equivalence testing, the following types will be used:

- Legal input values – Test values within boundaries of the specification equivalence classes. This shall be input data the program expects and is programmed to transform into usable values.
- Illegal input values – Test equivalence classes outside the boundaries of the specification. This shall be input data the program may be presented, but that will not produce any meaningful output.

The equivalence partitioning technique is a test case selection technique in which the test designer examines the input space defined for the unit under test and seeks to find sets of input that are, or should be, processed identically. The following table represents our equivalence classes, both valid and invalid.

<table>
<thead>
<tr>
<th>Input/Output Event</th>
<th>Valid Equivalence Classes</th>
<th>Invalid Equivalence Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input maximum number of allowed values</td>
<td>25 values</td>
<td>&gt; 25 values</td>
</tr>
<tr>
<td>Input integers</td>
<td>Integers between –999 and 999</td>
<td>Integers &gt; 999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Integers &lt; -999</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-integers (characters)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Non-integers (decimal values)</td>
</tr>
<tr>
<td>Load external file</td>
<td>Comma delimited file with only one value per line</td>
<td>No commas</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multiple entries per line</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No file content</td>
</tr>
<tr>
<td></td>
<td></td>
<td>File does not exist</td>
</tr>
<tr>
<td>Store external file</td>
<td>File exists</td>
<td>File does not exist</td>
</tr>
</tbody>
</table>

Boundary Value Testing

The acceptable range of values for this application was set by the development team. Due to the limitations of the GUI, the developers also limited the size of the input values to three digit integers. The valid and invalid ranges are shown below along with the corresponding valid and invalid boundary test values.

Acceptable Range: \(-999 \leq x \leq 999\)
Invalid Range: \(-\infty < x < -999\) and \(999 < x < +\infty\)

Valid Boundary Tests:
Invalid Boundary Tests:

Boundary_4: x = 1000
Boundary_5: x = -1000
Boundary_6: x = -999999
Boundary_7: x = 999999

Integration Testing

Incremental Testing

There are two primary modules that will need to be integrated: the Graphic User Interface module and the Tree Repository module (back-end). The two components, once integrated, will form the complete Binary Search Tree Application. The following describes these modules as well as the steps that will need to be taken to achieve complete integration. We will be employing an incremental testing strategy to complete the integration.

Module 1 - Graphic User Interface (GUI) Module

This module provides a simple GUI where the user can perform the different actions (functions). This module will be tested separate from the backend to check if each interface (e.g., insert button) is functioning properly, and in general, to test if the mouse-event actions are working properly. The testing will be performed by writing a stub for each element in the interface.

Module 2 – Tree Repository Backend Module

The “tree repository” provides the storage for the data elements and implements the algorithms and associated functionality of the binary tree. This module will be tested separate from the GUI by printing out the results to the Console. In testing this module we will follow the incremental testing method i.e. testing one function first and then keep adding additional function and test it again until all the required functions are tested.

When the GUI is combined with the backend module, we will have a complete binary search tree application. To achieve complete integration of these two modules, we will test each element in the GUI by replacing the stubs with the appropriate function from the back end. The results will be displayed within the GUI instead of through the Console. In testing the combined modules, we will follow the incremental testing method. Each stub will be replaced one at a time and tested. This will be done until all stubs have been replaced by the appropriate functions from the backend.

System Testing

The goals of system testing are to detect faults that can only be exposed by testing the entire integrated system or some major part of it. Generally, system testing is mainly concerned with areas such as performance, security, validation, load/stress, and configuration sensitivity. But in our case we will focus only on function validation and performance. And in both cases we will use the black-box method of testing.

Function Validation Testing
The integrated “Binary Search Tree Application” will be tested based on the requirements to ensure that we built the right application. In doing this test, we will try to find the errors in the inputs and outputs, that is, we will test each function to ensure that it properly implements the Binary Search Tree algorithms, and that the resulting tree displays the values in the proper location graphically. The behavior of each function, as well as their respective algorithms, are contained in the Software Program Specification.

<table>
<thead>
<tr>
<th>Function</th>
<th>Expected Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>see Software Program Specification</td>
</tr>
<tr>
<td>Store</td>
<td>see Software Program Specification</td>
</tr>
<tr>
<td>Insert</td>
<td>see Software Program Specification</td>
</tr>
<tr>
<td>Delete</td>
<td>see Software Program Specification</td>
</tr>
<tr>
<td>Search</td>
<td>see Software Program Specification</td>
</tr>
<tr>
<td>Clear</td>
<td>see Software Program Specification</td>
</tr>
<tr>
<td>List in Ascending Order</td>
<td>see Software Program Specification</td>
</tr>
<tr>
<td>List in Descending Order</td>
<td>see Software Program Specification</td>
</tr>
</tbody>
</table>

In addition, we will test:

- The interfaces to ensure they are functioning as desired (i.e. check if each interface is behaving as expected, specifically verifying the appropriate action is associated with each mouse_click event).
- The interaction between the GUI and the backend repository. In this case the data will be inserted and check if they are processed in the backend and give the expected output.

**Performance testing**

This test will be conducted to evaluate the fulfillment of a system with specified performance requirements. It will be done using black-box testing method. And this will be performed by:

- Storing the maximum data in the file and trying to insert, and observe how the application will perform when it is out of boundary.
- Deleting data and check if it follows the right sorting algorithm to sort the resulting data or output.
- Trying to store new data and check if it over writes the existing once.
- Trying to load the data while they are already loaded

**Entry and Exit Criteria**

This section describes the general criteria by which testing commences, temporarily stopped, resumed and completed within each testing phase. Different features/components may have slight variation of their criteria, in which case, those should be mentioned in the feature test plan. The testing phase also maps to the impact level definition when a defect is entered in the bug-tracking phase.

**Unit Testing**

Unit Testing is done at the source or code level for language-specific programming errors such as bad syntax, logic errors, or to test particular functions or code modules. The unit test cases shall be designed to test the validity of the programs correctness.
Black Box Phase

Black box testing typically involves running through every possible input to verify that it results in the right outputs using the software as an end-user would. We will use Equivalence Partitioning and Boundary Value Analysis complexity metrics in order to quantifiably determine how many test cases needed to achieve maximum code coverage.

Black Box Entry Criteria

The Black Box Entry Criteria will rely on the component specification, and user interface requirements. Things that must be done on entry to the Black Box stage:

- All Binary Tree functions, Load, Store, Clear, Sort Ascending, Sort Descending, Insert, Delete, Search, must either be coded or stubs created.
- The type of Black Box testing Methods will be determined upon entry. We will use Equivalency Partition, and Boundary Value Analysis.
- Equivalency Partition will include, Integer data types only, No Character data types accepted, each data field will be comma delimited, and there will be 1 value per line in the data file.
- Boundary Value Analysis will include, Integer data type values will have a boundary value of (-999,999). Zero is included. The file size is limited to 25 entries.

Black Box Exit Criteria

The Black Box Exit Criteria listed below explains what needs to be completed in-order to exit Black Box phase. To exit the Black Box phase 100% success rate must be achieved. Things that must be done upon exiting the Black Box stage:

- The Equivalence Classes will have been created for the valid and invalid input values. For our Binary Tree program the input domain values for Equivalence Partitions will include Integer data types only, each data field will be delimited by a comma and carriage return, and one data value per line in the input data file.
- The Equivalency Partition Method will have generated Test Cases based on the Equivalence classes. The invalid input domain values for Equivalence classes will include loading an empty input data file, inputting character strings, entering a delimiter other that a comma, and entering more than one data value per line in the input data file.
- Boundary Value Analysis will have generated Test Cases based on the boundary values of Integer data type values of (-999,999). These Test Cases will test for values above and below the specified boundary values. For example, values that include infinity, negative infinity, zero, and decimal numbers.
- Another set of Test Cases will have been created based on the boundary value of the file size limited to 25 entries. These Test Cases will test for zero entries in the data input file, and greater than 25 entries.
- All code bugs that are exposed are corrected.

White Box Phase

The White Box criteria apply for purposes of focusing on internal program structure, and discover all internal program errors. Defects will be categorized and the quality of the product will be assessed.

White Box Entry Criteria

The White Box Entry Criteria will rely on the QA engineers verifying that the major features work alone but not necessarily in combination; exception handling will not be implemented. The design and human interface are stable. Things that must be done on entry to the White Box stage:
• All Binary Tree functions, Load, Store, Clear, Sort Ascending, Sort Descending, Insert, Delete, Search, must be coded.
• The type of White Box testing Methods will be determined upon entry. We will use Basis Path Testing and Function Validation testing on all Binary Tree properties.
• Black Box Testing should be in its late stages.

After the White Box criteria have been met, the product enters the White Box stage. During White Box stage Development Engineering’s emphasis is on refining the product and fixing defects. Information Design’s emphasis is on developing product user documentation.

**White Box Exit Criteria**

The Binary Tree in the White Box stage should have a generally stable feel to it. White Box testing continues until the Black Box or next milestone criteria are met. To exit the White Box phase 100% success rate must be achieved. The following describes the state of the product upon exit from the White Box Stage:

• All Binary Tree functions, Load, Store, Clear, Sort Ascending, Sort Descending, Insert, Delete, and Search are implemented, operational and tested.
• All Branch Testing test cases will be generated. The test cases will be generated from the Control Flow diagrams of all functions.
• The Binary Tree graphical interface has been reviewed and found to satisfactory by development Engineers, and QA Engineers, and is stable, that is, no further changes to dialog boxes or other interface elements are planned. Minor changes (word-smiting, etc.) are acceptable, but must be arranged with the Development and Test Engineers.
• All code bugs that are exposed are corrected.

**Integration Test**

There are two modules that will be integrated for Integration Testing. The two modules are The Graphic User Interface module and the Tree Repository module (back-end). The two components will consist of a mixture of stubs, driver, and full function code. The following describes the entry and exit criteria for Integration testing.

**Integration Test Entry Criteria**

The Integration Test Entry Criteria will rely on both modules to be operational. The Binary Tree design and human interface must be stable. Things that must be done on entry to the Integration Test stage:

• All Binary Tree functions, Load, Store, Clear, Sort Ascending, Sort Descending, Insert, Delete, Search, must either be coded and/or stubs created.
• The Graphical User Interface must either be coded and/or a driver and stubs must be created. The driver is implemented to facilitate test case input and output values.
• Interfaces and interactions between the Binary Tree Module and the Graphical User Interface must be operational.
• A bottom-up Integration Test Strategy will be conducted. The low level details of the Binary Tree and graphical interface will be integrated. A driver will be written to facilitate test case input and output values. The driver will temporarily satisfy high-level details of the input and output values.
• Black Box Testing should either be in its late stages or completed.
• White Box Testing should have begun.

**Integration Test Exit Criteria**
The Integration Test Exit Criteria will rely on both modules to be operational. The Binary Tree design and human interface must be stable. To exit the Integration Testing phase 100% success rate must be achieved. Things that must be done on exit from the Integration Test stage:

- All code bugs that are exposed are corrected.
- The Binary Tree Module and Graphical User Interface Module will interact together with complete accuracy, according to the System Specification Design. All discrepancies are corrected.
- Both Modules are ready for System Testing. Stubs and drivers are replaced with fully functional code.
- Black Box Testing is completed.
- White Box Testing should either be in its late stages or completed.

**System Test**

The System Test criteria apply for purposes of categorizing defects and the assessing the quality level of the product. All elements of the Binary Tree Module and Graphical User Interface are meshed together and tested as a whole. System test focuses on functions and performance, reliability, installation, behavior during special conditions, and stress testing.

**System Test Entry Criteria**

The Entrance Criteria specified by the Development Engineers, should be fulfilled before System Test can commence. In the event, that any criterion has not been achieved, the System Test may commence if both Development and Test Engineers are in full agreement that the risk is manageable.

- The Graphical User Interface and the Binary Tree back-end Module must be fully functional.
- All developed code must be unit tested. Unit and Link Testing must be completed and signed off by the development team.
- All test hardware and environments must be in place, and free for System test use.
- All Black Box testing must be complete and exposed bugs must be corrected.
- All White Box testing must be complete and exposed bugs must be corrected.
- Integration Testing must be complete and exposed bugs must be corrected.
- Function Validation Testing is the accepted method of testing for all Binary Tree functions: Load, Store, Clear, Sort Ascending, Sort Descending, Insert, Delete, and Search. The Graphical User Interface will be the method of interacting with the system, so the GUI will be tested thoroughly.
- Development and Test Engineers agree that Function Validation Testing will cover function performance, reliability, stress and load testing.

**System Exit Criteria**

The Exit Criteria must satisfy all the criteria listed below. This verifies that all elements of the project mesh properly. This is to make sure that all the system functions and performs according to the System Specification Document.

- All Function Validation Testing is 100 percent successful. Testing for all Binary Tree functions: Load, Store, Clear, Sort Ascending, Sort Descending, Insert, and Delete, and Search interact with complete accuracy.
- No degradation of System performance across different platforms of Windows operating system will be affected. (Windows 95 or above is acceptable)
- The Graphical User Interface performs to System Specification Requirements.
- All the Binary Tree properties are expressed correctly through the Graphical User Interface.
- All input fields on the Graphical User Interface are working correctly.
- All high priority errors from System Testing must be fixed and tested.
• If any medium or low-priority errors are outstanding – the Development Engineers and Test manager must sign off the implementation risk as acceptable.

**Shipping or Live Release**

The Binary Tree testing is scaled down and combines all phases of testing into two phases – Function Complete and Regression testing – and follows the release criteria.

**Shipping/Live Release Entry Criteria**

The criteria for entering the final stages are as follows:

- QA verifies that all open product defects, regardless of fixed defects, documented, deferred, or otherwise addressed.
- QA verifies that regression testing on all product defects and the entire product has been completed.
- QA verifies that all bugs “For Verify” have been regressed.

The software is frozen when the product passes its final milestone. If any code changes are made after the final milestone, the features fixed must be re-tested. QA, and Development Engineers closely monitor fixes that go into the final build to minimize risk. After the final milestone criteria have been met, the product enters the Live Release stage.

**Shipping/Live Release exit Criteria**

The Shipping/Live Release stage is when the product is ready for general availability to the public and the user documentation is final. The product must fully satisfy its release specifications and the user documentation must adequately describe the product’s functionality. Both should be ready for use by the end user.

- QA tests the final product version to verify that the product to be released to the general public is of the utmost quality and satisfies original design specifications.
- The product must receive approval from the product team.
- QA and Development must prepare Release Notes.

The product is now ready to ship or published to production environment.

**Bug Tracking/ Bug Process**

During testing, the testing team members normally encounter behavior that goes against a specified or implied design requirement in the product. When this happens, we will document and reproduce the bugs for the developers.

**Expectation of a bug:**

- Keep track of what version of the application the bug is found
- Determine if bug has already been written up
- Indicate the steps to reproduce the bug – write enough details for others looking at the bug to be able to duplicate it; exclude unnecessary steps (i.e. If access point is irrelevant, be more general in your steps).
- Actual results – be specific on your findings.
- Expected results – how the product should behave based on the specified or implied requirements.
- Implications – How does the defect affect the quality of the product?

The following chart defines the impact levels to be used when entering bugs.

<table>
<thead>
<tr>
<th>Impact</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – Fatal</td>
<td><strong>Test Stopper:</strong> If you can’t access a function and need the bug to be fixed immediately. The defect prevents QA from testing the feature area, sub-area or functionality of the feature.</td>
</tr>
<tr>
<td>2 – Serious</td>
<td><strong>Beta Stopper:</strong> This is a bug that users would experience such as: data corruption, calculation errors, incorrect data, UE’s and system crash on common user scenarios, significant QA risk, and major UI defects.</td>
</tr>
<tr>
<td>3 – Minor</td>
<td><strong>Live Release:</strong> A bug that must be fixed before the product is officially completed, UE’s or crashes, content, and UI and graphic changes required for release.</td>
</tr>
</tbody>
</table>

**Various Roles in Bug Resolution**

- **Author** – The person who wrote the bug; this will be someone on the QA team
- **Resolver** – Normally an Engineer assigned to a specific area of the application.
- **Verifier** – normally a QA Engineer responsible for testing the fix and closing the bug.
Bug Report Form
Roles and Responsibilities

Development Team

Code Development Project Leader – V. Stanton

- Ensure Phase 1 is delivered to schedule and quality
- Ensure exit criteria are achieved prior to system test signoff
- Regularly review testing progress with test controller.
- Raise and manage issues/risks relating to project or outside test teams control.
- Review and sign off test approach, plans and schedule.

SQA Project Leader – H. Frezghi

- Ensure Phase 1 is delivered to schedule and quality
- Regularly review testing progress
- Manage issues/risks relating to System Test Team
- Provide resources necessary for completing system test

Testing Team

Test Planner / Controller – N. Monge

- Ensure Phase 1 is delivered to schedule and quality
- Produce high level and detailed test conditions
- Produce expected results
- Report progress at regular status reporting meetings
- Co-ordinate review and signoff of test conditions
- Manage individual test cycles and resolve tester queries/problems.

Lead Tester – T. Wilkinson

- Identify test data
- Execute test conditions and mark-off results
- Prepare software error reports
- Administrate error measurement system
- Ensure test systems outages/problems are reported immediately and followed up.
- Ensure entrance criteria are achieved prior to system test start.
- Ensure exit criteria are achieved prior to system test signoff.

Test Schedule

The section contains the overall project schedule. It discusses the phases and key milestones as they relate to quality assurance. It discusses the testing goals and standards that we’d like to achieve for each phase of testing that will be deployed, e.g., Usability Testing, Code Complete Acceptance, Beta Testing, Integration Testing, Regression Testing, System Testing.

The key dates for overall Binary Tree development and Testing are outlined below. For details on the schedule, refer to the Binary Tree Project Schedule (this document). For details on general Engineering QA deliverables, refer to the test plan document.

<table>
<thead>
<tr>
<th>Binary Tree Program</th>
<th>End Date</th>
<th>Notes</th>
<th>QA Deliverables/Roles</th>
</tr>
</thead>
</table>


<table>
<thead>
<tr>
<th>Milestones</th>
<th>Date</th>
<th>Description</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning Phase</td>
<td>02/20/15</td>
<td>At this Milestone, the high level planning should be completed. Some of the deliverables are: Project Plan, Program function specifications.</td>
<td>High-level test planning activities, which include preliminary development of Master QA Plan (this document, QA schedule).</td>
</tr>
<tr>
<td>Design Phase</td>
<td>02/27/15</td>
<td>This is a feature-driven milestone where the requirements and initiatives are further defined and solutions are finalized. The deliverables for this phase are Program source code and other design related documents.</td>
<td>Development and Test engineers participate actively in feature design by inspecting and reviewing the requirements and design documents. As the design documents are completed, the test engineers are encouraged to start working on the Test Plan document and test design planning.</td>
</tr>
<tr>
<td>Code Complete -Infrastructure</td>
<td>03/06/15</td>
<td>This milestone is when all infrastructure development and functions should be complete. The testing team should have preformed unit &amp; integration testing before checking the code into any build.</td>
<td>The Test Engineers should have completed or in the final stages of their preliminary Infrastructure Test Plan, test cases and other QA documents related to test execution for each feature or component such as test scenarios, expected results, data sets, test procedures, scripts and applicable testing tools.</td>
</tr>
<tr>
<td>Code Complete -Function</td>
<td>03/10/15</td>
<td>This milestone includes unit testing and code review of each function component prior to checking the code into the test phase. The deliverables include system-testing specification, Unit testing specifications, Integration plan.</td>
<td>The Test Engineers should have provided Code Complete Assessment Test to Development Engineer one week prior to Code Complete Review date. The Test Engineers should also have completed or in the final stages of their preliminary White Box Test Plan, test cases and other QA documents related to test execution for each feature or component such as test scenarios, expected results, data sets, test procedures, scripts and applicable testing tools.</td>
</tr>
<tr>
<td>Beta Ready</td>
<td>03/15/15</td>
<td>This milestone represents that all features are ready for Beta release shutdown.</td>
<td>2 Weeks regression of Binary Tree features to Beta and preparation for Beta Shutdown.</td>
</tr>
<tr>
<td>Feature Complete</td>
<td>03/15/15</td>
<td>This phase allows for feature clean up to verify remaining bug fixes and regression testing around the bug fixes. This milestone indicates that the feature is ready for Beta regression.</td>
<td>All bugs verified and QA documentation is finalized. The test Engineers should assess that Binary Tree features are ready for Beta regression and have started their preliminary Test Summary Reports.</td>
</tr>
<tr>
<td>Regression Test</td>
<td>03/20/15</td>
<td>This milestone represents that all Binary Tree code and GUI interface to the Binary Tree is ready for Regression Testing.</td>
<td>Complete regression test execution of complete system and update Test Summary Reports for regression.</td>
</tr>
</tbody>
</table>
The Microsoft Project schedule is included at the end of this document.

**Deliverables**

- Program function specifications
- Program source code
- Test plan document - this document should address testing objectives, criteria, standards, schedule and assignments, and testing tools.
  - Unit Testing Plan
  - Integration Plan
  - System Testing Plan
- Test Design Document
  - Unit white-box test design – covers white testing criteria, methods and test cases
  - Unit black-box test design – covers black-box testing criteria, methods and test cases
  - System test design – covers system test criteria, methods, and test cases, scripts.
- Test report document
  - Unit white-box test report – covers unit white box test results, problems, summary and analysis
  - Unit black-box test report – covers unit black box test results, problems, summary and analysis
  - System Test report – covers system test results, problems, summary and analysis
INTRODUCTION

Purpose
The purpose of this document is to identify and describe a configuration management (CM) process for PWL. This plan describes in simple, straightforward terms the processes required to ensure that the inevitable network changes occur within an identifiable and controlled environment.

This document is intended to be a living document. As implements the components of this plan, and to facilitate the every changing state-of-the-art, the SN CM process may need to be refined. Consequently, the final version of this document should itself be placed under configuration management and the respective changes managed accordingly.

Background

The IT organization has identified the need for a configuration management plan to control changes to the SN. During emergencies, the SN must be rapidly reconfigured to support the establishment of Field Offices. Currently, informal operational processes followed by each entity involved with the SN are the primary means of controlling SN changes. These processes are frequently undocumented; consequently, the IT organization cannot determine the status of current SN architecture, network component configuration and proposed changes. This plan addresses this deficiency. It establishes a consistent, cross-organizational CM process for the SN architecture and its components. It provides both SN managers and technical personnel the information they need to implement the SN CM activities and their flow.
Scope

The scope of this document is the identification of a top-level configuration management plan for the SN. This plan presents CM activities for the data portion of LAN/WAN (e.g., switches, routers, and hubs). Specifically excluded from this plan are network server hardware and operating systems. Section 2 describes the network scope in greater detail.

Document Overview

This document is divided into five (5) major sections, the Introduction, the System Overview, the Configuration Management Components, the Configuration Management Process, and Next Steps. The organization of this document is as follows:

- **Section 1.0 – Introduction**: This section presents preliminary information concerning this document. The scope of this document, the organization of this document, and any references utilized in the assembly of this document.

- **Section 2.0 – System Overview**: This section presents an overview of the SN architecture.

- **Section 3.0 – Configuration Management Components**: This section describes the components of the configuration management process.

- **Section 4.0 – Configuration Management Process**: This section describes the configuration management process to be used.

- **Section 5.0 – Next Steps**: This section presents recommended activities to ensure successful implementation of the SN CM process.

**References**

- The following references were used in preparation of this document:

**CONFIGURATION MANAGEMENT COMPONENTS**

This section describes the components of the CM process. These components are presented using the conventional CM framework – configuration identification, configuration change control, configuration status accounting, configuration reviews. Section 4 presents how these components will be used in the SN CM process.

Configuration management involves identifying the configuration of a network system at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the lifecycle. The items placed under configuration management include the software and hardware products (e.g., routers and switches) that comprise the network as well as items required to create or maintain these products (e.g., initial routing tables and switch configuration data). Proper configuration management enables an organization to answer the following questions:

- What is the process for making changes to the network?
- Who made a change to the network?
- What changes were made to the network?
- When were the changes made?
- Why were the changes made?
- Who authorized the changes?
Organizations and Responsibilities

The following organizations in the Information Technology (IT) Services Division will be involved in configuration management activities for the:

- Architecture Group
- Operation Group
- Deployment Group

In addition, two CM entities – the SN Change Control Board and the SN Working Group will provide cross-organizational CM control and coordination.

IT Services Directorate

Architecture Group is responsible for the SN architecture and its components. It tests and deploys SN components, modifies existing network components, and identifies potential SN enhancements. Only Architecture Group is authorized to make changes to the SN and its components. Architecture Group will be responsible for the establishment of the Component Folders.

Operation Group is responsible for the day-to-day operation of the SN. It maintains the deployed components of the SN and ensures that the SN is up and operational. The Operation Group Help Desk is the initial point-of-contact for reported network incidents.

Deployment Group is responsible for deploying field office network hardware and integrating them into the SN.

SN CM Entities

The SN Change Control Board (CCB) and the SN Working Group (WG) will provide cross-organizational CM control and coordination.

SN Change Control Board

The SN CCB will be established as a formal approval authority for changes. It primarily exists to control changes to the SN architecture (e.g., deployment of a new piece of hardware); however, any SN issue with significant cross-organizational impact should involve the SN CCB. Written Change Requests (CRs), Impact Assessments (IAs), and meeting minutes will document the CCB’s consideration of an issue.

A Change Request (CR) form will initiate CCB consideration of a change. Typically, Architecture Group will originate most CRs, but any SN organization may submit a CR to the CCB. Upon receipt of a CR, the CCB will request each member organization to research the incident and record its analysis on an IA. These IAs will document the data upon which the SN CCB will have based its decision. The SN CCB will identify the appropriate change resolution activities and will authorize the affected organizations to perform them. The SN CCB will be notified immediately when the actual change has been completed and the configuration status information will be updated accordingly. The decision of the SN CCB and its rationale will be recorded on the CR. The CR will be closed and copies distributed to each member organization. Section 4 presents this process in greater detail. Appendix A contains sample Change Request and Impact Assessment forms.

The SN CCB will be comprised of representatives from the following IT organizations:

- Architecture Group
- Operation Group
- Deployment Group

Each representative must have the authority to commit his or her organization. Since the SN CCB will consider and resolve cross-organizational issues, its Chair should be a person with the authority to mandate actions to the IT organizations. In addition, since final responsibility resides with the CCB Chair, this person will have the authority to mandate or override decisions made by the CCB. Typically, the CCB Chair is a representative from the next level of management. Figure 3-1 graphically presents the organization of the SN CCB. Appendix B contains a template for the SN CCB charter.
**SN Working Group**

The SN WG is a less formal change control organization that coordinates minor SN changes (e.g., changing settings on a particular router) between Architecture Group and Operation Group. The SN CCB delegates this authority to the SN WG to provide a more streamlined CM control mechanism for those changes that do not affect multiple SN organizations. Although the WG is less formal than the CCB, all requests and decisions must still be documented.

An Engineering Request (ER) form submitted to Architecture Group will initiate WG consideration of a change. Typically, Operation Group will originate most ERs, but any SN organization may submit an ER. Upon receipt of an ER, Architecture Group will research the request and record its analysis on the ER. If the request only involves component-level changes, then WG will authorize the changes and the ER will document the decision. If the incident involves more significant architectural changes, then the WG will originate a CR, and the CR Control Number will be recorded on the ER for traceability. The ER will be closed and returned to the originator. Section 4.0 presents this process in greater detail. Appendix A contains a sample Engineering Request form.

The SN WG will consist of the managers of the Architecture Group and Operation Group organizations or their designated representatives. The SN WG may be considered a mini-CCB.

**Configuration Identification**

Configuration identification (CI) involves identifying the components of the network, uniquely identifying the individual components, and making them accessible in some form. A proper configuration identification schema identifies each component of the network and provides traceability between the component and its configuration status information. Proper configuration identification answers the following questions:

- What is the configuration of the network?
- What are the components of the network?
- What are the versions of the network components?

The major activities of configuration identification are:

- Selecting network components to be placed under CM control
- Creating an identification scheme for the components to uniquely identify each individual component

The following sections present the configuration identification activities for the SN.

**Select Network Components**

This plan addresses the CM of the SN network infrastructure – switches, routers, and hubs. Specifically excluded from this plan are network server hardware and operating systems.
Architecture Group, in conjunction with the CCB, will determine the network hardware components to be placed under CM. The following is a preliminary list of SN network hardware items to be controlled:

- Cisco 7000 Series Routers
- Cisco 5000 Series Routers
- Cisco 4000 Series Routers
- Cisco 3000 Series Routers
- Cisco 2500 Series Routers
- Ascend MAX Inverse Multiplexer
- I-MUX Dialup
- CDDI/FDDI Concentrators
- Networth 4000 Concentrators
- Bay Stack Hubs
- Acculan Hubs

Architecture Group, in conjunction with the SN CCB, will determine the network software components to be placed under CM. The following is a preliminary list of the SN network software items to be controlled:

- HP Open View

**Uniquely Identify Each Component**

Applying CI naming involves setting naming standards based on criteria about the component's location, function, etc. An example may include naming a router based on the location, model, and function, such as B9-C7000-FIN (for this example, a CISCO 7000 router located in building 9, used for Financial communication). The key in CI naming is setting a usable naming standard that can be applied across the entire enterprise of components within the CM scope.

**Configuration Change Control**

Configuration change control involves controlling and managing the changes to the network. The goal of change control is to establish mechanisms that will help ensure the integrity of the network. Proper configuration change control answers the following questions:

- What network components are controlled?
- How are changes to the network controlled?
- Who controls the changes to the network?

The major activities of configuration change control are:

- Defining and documenting the change control process
- Identifying and maintaining network configuration baselines
- Identifying and controlling network changes

The following sections present the configuration change control activities for the SN.

**Define the Change Control Process**

At a high-level, the SN change control process consists of the following basic steps:

- Identifying and classifying a change to the network
- Evaluating what components in the current network configuration needs to be changed
- Testing or modeling the impact of the change upon the current network
- Implementing the change if it is approved.

Figure 3-2 graphically presents the phases and the high-level activities in this process. A more detailed description of each phase and a decomposition of each step's activities are presented in Section 4.0.
FIGURE 3-2. SN CM PROCESS OVERVIEW

Maintain Network Configuration Baselines
A configuration baseline is the foundation of configuration management. Each baseline captures an approved snapshot of the SN and its components at a given point in time. The conventional configuration management model constructs a system baseline from the top-down. In this single-tiered model, a baseline is comprised of a specific release of each component and any change to any component must be considered and approved by the CCB. However, since the SN is an extremely dynamic network, this top-down model would be very costly to implement and maintain; therefore, the SN will be managed using a two-tier CM model.
At the first-tier SN-architecture level, baseline control will be centralized at the CCB. The CCB will baseline a basic network architecture, e.g., FDDI backbone, Ethernet, Cisco routers, etc., and the CCB must consider and approve any changes to that architecture, e.g., deploying an Ascend router. This will prevent significant network changes from being performed before all affected organizations have been informed and have provided their input.

At the second-tier SN-component level, baseline control will be more decentralized. A component baseline will be established for each SN component that will capture the operational parameters with which that component was evaluated and deployed. Any changes to this baseline, e.g., updating the routing parameters, must be approved by Architecture Group and documented by the person making the changes. Typically these changes will not require CCB approval, but periodic configuration reviews will enable the CCB to monitor component level changes and refine the process if necessary.

This two-tier CM model is shown in figure 3-2 above. This model will ensure that significant network changes to the SN-architecture are performed in a controlled, well-documented manner, while still allowing network engineers to react promptly to ensure effective operation of the network components.

Control Network Changes

Generally, Architecture Group, in accordance with the SN CCB, will control first-tier network architecture changes; while operational procedures will control most second-tier network component changes.

SN Architecture

At the SN-architecture level, only the SN CCB will have the authority to approve and make changes to the components of the SN architecture baseline. Before considering any change, the potential change must be documented using a SN Change Request (CR) form. This form describes the desired change, the justification for the change, the impact of implementing the change, and the eventual CCB decision regarding the change. CRs normally document network changes originating from Architecture Group, but any SN engineer or organization may submit a CR.

The CR will be submitted to the Chair of the CCB who will route the CR to each organization on the SN CCB. Each organization will fill in a SN Impact Assessment (IA) form documenting the impact of the change upon their organization. If an organization will be affected by the change, the IA will be returned with notations or comments documenting the organization’s technical analysis, the impact on the organization’s schedule, cost, and labor resources, and any suggested alternatives to the proposed change. If an organization will not be affected by the change, the IA will be returned indicating "no impact".

The CCB will meet either in person or by teleconference to discuss the change. The CCB will use the IAs and the meeting discussion to decide whether to approve or disapprove the proposed change. The CCB meeting minutes will clearly document its decision and the rationale supporting its decisions. These CCB minutes will be distributed to all CCB organizations and retained in the SN project files.

Figure 3-3 graphically presents the SN CCB process. Appendix A contains sample SN Change Request and SN Impact Assessment forms.
SN CCB Receives CR

CR Originated By Other FSN Organization

CR Originated By Arch Group

CCB Approves or Disapproves CR

Arch Group Impact Assessment

Op Group Impact Assessment

CCB Directs Arch Group To Research CR

Arch Group Impact Assessment

CCB Closes CR
FIGURE 3-3. SN CCB PROCESS

**SN Components**
At the SN-component level, a larger number of persons will be able to make changes. Architecture Group, in accordance with the CCB, will identify the specific SN personnel authorized to make changes and the specific changes that each person is authorized to make. The CR process will be used to request and approve changes to this authorization list, but no IAs will be required.

Each SN component will have an associated SN Component Folder (CF) which will be used to track component level changes. Any change, no matter how minor, made to that component must be recorded in the CF. The CF will be started when a component is deployed into the SN and will depict the initial configuration of the component. Subsequent changes to that component will be recorded by the entries in the CF. The CF pertaining to a component may be either hardcopy or softcopy -- the critical factor is that the CF be readily available to network personnel. (If it is difficult to enter and update information in the CF, the CF becomes a hindrance rather than a tool and probably won’t be maintained).
Configuration Status Accounting

Configuration status accounting involves the recording and reporting of the change process. The goal of configuration status accounting is to maintain a status record of all items in the network baseline, thus providing traceability of all changes to the network. Proper configuration status accounting answers the following questions:

- What changes have been made to the system and when were they made?
- What components were affected by this change?

The major activities of configuration status accounting are:

- Identifying the configuration status information to be recorded
- Maintaining a record of configuration changes
- Reporting the status of network configuration management
- The following sections present the configuration status accounting activities for the SN.

Identify Status Tracking Information

- The following configuration status information will be regularly tracked at the SN-architecture level:
  - Current SN Network Architecture
  - Inventory of Current Network Components
  - Open Change Requests
  - Closed Change Requests
  - Components Affected By Change Requests
  - Open Engineering Requests
  - Closed Engineering Requests

This information will enable the IT organization to monitor the SN architecture, to identify potential troublespots, and to plan potential enhancements.

At the component-level, the CFs will provide configuration status information for each individual SN component. This information will not normally be tracked at the organizational level, but can and should be made available in special circumstances.

Maintain Status Tracking

First-tier, architectural configuration status information will be recorded and maintained in a centralized repository; most typically some sort of database. This repository will contain the inventory of network components and their status, as well as the proposed and implemented changes to the network. Second-tier, component configuration status information will be recorded and maintained in the individual Configuration Folders. These folders may be hardcopy (e.g., a manila folder next to a hub) or electronic (e.g., a database containing all SN routers and their configuration). Regardless of their form, CFs must be easy for network personnel to access so that they may record what they did to the component at the same time they performed the action.

Since the configuration status information is crucial to ensuring the integrity of the SN baseline, its components, and the SN CM process, only authorized personnel should update this data. The SN CCB will identify the specific SN personnel authorized to update the status information changes and the specific changes that each person is authorized to make.

Report Status Tracking

Regular monitoring of the configuration status information will enable IT to identify trends and potential troublespots in the SN. Each IT organization and the SN CCB will determine the reports required for performing this analysis.
Configuration Reviews

Configuration reviews will be performed periodically to verify the correctness of the configuration status accounting information. The goal of a configuration review is to verify that all network components have been correctly identified and that all network changes have been properly managed. Periodic configuration reviews will also enable to assess the effectiveness of the SN CM process and to identify potential modifications. Proper configuration reviews answer the following questions:

- Are the configuration status accounting records accurate and complete?
- Does our configuration management process work effectively?

The major activities of a configuration review are:

- Identifying the information to be reviewed and performing the review
- Documenting and analyzing the results of the review
- The following sections present the configuration status accounting activities for the SN.

**Perform Configuration Review**

- Periodically, the configuration status information will be reviewed to verify its accuracy and completeness. The SN CCB will identify the configuration reviews to be performed. The following is a preliminary list of configuration reviews:
  - Physical Network Review

The Physical Network Review will analyze and note any discrepancies between the configuration status information and the physical SN network. This is a detailed review which compares the data in the repository and CFs with the actual physical configuration of the deployed network components. This review is roughly equivalent to the Physical Configuration Audit in traditional CM.

**Document and Analyze Results**

The results of the configuration review will be documented and made available to the SN CCB and SN IT organizations. These organizations will use the findings to identify and correct discrepancies in the configuration status information. In addition, the SN CCB should analyze inefficiencies and problems identified in the SN CM process and undertake to resolve them.

**CONFIGURATION MANAGEMENT PROCESS**

This section presents the SN configuration management process. This process describes verbally and pictorially the CM activities and their flow. The activity descriptions provide adequate information to enable SN personnel to make CM decisions, but they are not written at the procedure level of detail.

**Process Overview**

The SN CM process consists of four phases:

- Classification – An incident or change is identified and routed to the appropriate organization for resolution
- Evaluation – An initial solution is identified. CM control is provided by either the SN Change Control Board (CCB) or Working Group (WG) depending on the nature of the proposed solution (component-only or SN architecture).
- Modeling and Testing – If the proposed solution involves an architectural change, the change is modeled and tested to determine its effect on the existing SN architecture.
- Implementation – The final solution is approved and deployed to the SN.
Classification is a triage phase. It begins when a SN incident or change is identified; typically, when the Help Desk receives a trouble report. Operation Group evaluates the report and determines whether it can resolve the incident itself (e.g., a modem needs to be reset) or whether the solution requires a change to the SN (e.g., a router configuration needs to be modified). If Operation Group resolves the incident itself, it updates the Component Folder (CF) to record its actions and the report is closed. If Operation Group cannot resolve the incident or it requires a change to SN, Operation Group originates an Engineering Request (ER) which formally transfers the incident to Architecture Group for resolution.

Evaluation is an analysis phase. It begins when Architecture Group receives an ER request from Operation Group (or any other SN organization) documenting an SN incident. Architecture Group analyzes the incident and identifies a potential solution. If the solution only involves minor changes to existing SN components, then Architecture Group presents the incident and solution to the SN WG for approval. If the solution involves significant component changes or SN architectural changes, then Architecture Group originates a Change Request (CR) and presents it to the SN CCB.

Modeling and Testing is a research phase. It begins when a CR is under consideration by the SN CCB. Architecture Group will model and test the potential solution to determine its impact on the SN. Other SN organizations will research the potential solution to determine its impact on their organizations. Each organization will document the results of its research using an Impact Assessment (IA). The SN CCB will use these IAs as the basis for approving or disapproving the potential solution.

Implementation is a deployment phase. It begins when the SN CCB or WG approves a solution. Architecture Group will implement the change and deploy it. If the change performs properly, Architecture Group creates and/or updates the Component Folders (CF) to record its actions and the ER or CR is closed.

The following sections discuss each phase in greater detail.
Classification

Classification begins when a SN incident or change is identified; typically, when the Help Desk receives a trouble report. The incident or change is documented in the trouble ticket system. Operation Group evaluates the trouble ticket to determine whether it can resolve the incident itself or whether the solution requires a change to the SN. If Operation Group can resolve the incident itself and is authorized to do so (e.g., a modem needs to be reset), then Operation Group solves the incident, records its actions in the CFs, and closes the trouble ticket. If Operation Group cannot resolve the incident or is not authorized to do so (e.g., a router configuration needs to be modified) then it originates an ER and formally transfers the incident to Architecture Group for resolution. The flow then enters the Evaluation phase.

Figure 4-2 presents the activity flowchart for the Classification phase.

Evaluation

Evaluation begins when Architecture Group receives an ER request from Operation Group (or any other SN organization) documenting an SN incident. First Architecture Group determines if the ER is a valid request; then it analyzes the ER to determine the extent of its impact. Not all SN changes are major (e.g., replacement of a defective hub with a good hub of the same make and model) and may need only Architecture Group to effect the change. These minor changes will proceed directly to the Implementation phase. If the ER may require significant changes to the SN architecture or its components (e.g., a router needs to be reconfigured), then Architecture Group determines whether the potential solution requires SN architectural-level or component-level changes. A potential solution may require a combination of architectural and component-level changes; however if the potential solution involves any architectural considerations, then the entire incident shall be considered an architectural change.

For architectural-level changes, Architecture Group originates a CR and presents it to the SN CCB. Architecture Group also performs preliminary research to assist the CCB in determining whether the change is plausible. If the CCB disapproves the potential change, then the CR is closed and the flow ends; if the CCB decides that the potential change should be implemented, then the flow proceeds to the Implementation phase;

For component-level changes, Architecture Group performs preliminary research which it then presents to the SN WG. The WG reviews the incident and potential solution and determines whether the change is plausible. If the WG disapproves the potential change, then the CM process is exited; if the WG decides that the potential change should be implemented, then the flow proceeds to the Implementation phase.

Figure 4-3 presents an activity flowchart for the Evaluation phase.
Modeling and Testing

Modeling and Testing begins when a potential solution is under consideration by the SN CCB or WG. Architecture Group will develop and document a modeling or testing strategy, configure the evaluation environment, and execute the test. Data from the test will be analyzed and the impact on the SN documented using an IA. The Other SN organizations will also research the potential solution to determine its impact on their organizations. These organizations will also document their research using IAs.

The IAs form the basis of the SN CCB’s decision regarding the potential solution. The SN CCB will consider the test results, the impact on the SN, and any organizational effect and either approve or disapprove the solution. If the solution is approved, then the flow proceeds to the Implementation phase. If the solution is disapproved, then Architecture Group reevaluates the incident and potential solutions and the flow returns to the Evaluation phase.

Figure 4-4 presents an activity flowchart for the Modeling and Testing phase.

Implementation

Implementation begins when the SN CCB or WG approves a solution. Architecture Group will plan, execute, and deploy the change. Architecture Group will monitor, verify, and document that the change produces satisfactory results.

If the change performs properly, Architecture Group creates and/or updates the Component Folders (CF) and other SN documentation to record the new baseline. If SN personnel require new skills to support the change, Architecture Group will conduct training. The SN CCB will close the ER or CR and Operation Group will assume operational maintenance. The flow will end. If the change does not perform properly, then Architecture Group reevaluates the incident and potential solutions and the flow returns to the Evaluation phase.

Figure 4-5 presents an activity flowchart for the Implementation phase.

FIGURE 4-4. MODELING AND TESTING PHASE ACTIVITIES

FIGURE 4-5. IMPLEMENTATION PHASE ACTIVITIES

NEXT STEPS

This section presents recommended activities to ensure successful implementation of the configuration management process described in this plan.

Plan SN CM Implementation

Implementation of the SN CM process requires numerous activities to be conducted across SN organizations. The SN CM Implementation Plan will help ensure that these transition activities are completed and that each SN organization is ready for transition to the SN CM process. In addition, this plan
will help ensure that the transition is performed in an orderly and controlled fashion.

The SN CM Implementation Plan will:

- Identify the implementation strategy
- Identify the implementation activities for the SN organizations
- Identify the implementation schedule
- Identify training requirements

IT has already tasked development of this plan as a part of this task order.

**Finalize SN Component Inventory**

The SN component inventory forms the basis for the SN architecture baseline and the SN Component Folders. This inventory defines the scope of the entire CM process, therefore it is critical that this inventory be accurate and complete.

IT has already begun this activity. The Equipment List, and the Master Inventory of Wide Area Network Routers, provides a good basis for this inventory. IT should finish identifying SN network components and begin populating a centralized repository with the information.

**Establish SN Component Folders**

The SN Component Folders are an integral part of the SN CM process. Each SN component should have a CF identifying its configuration at deployment, its current configuration settings, and any actions performed on the component. Since the CFs are such a critical component of the SN CM process, the CFs must be in place before transitioning to this CM process.

**Identify Candidate CM Tools**

CM tools vary greatly in their capabilities and robustness. Some CM tools only provide version control (e.g., Unix’s SCCS); while other tools provide built-in workflow support for the entire CM process (e.g., PCMS). Before selecting any tools to support the SN CM process, IT should identify the areas of the SN CM process that would benefit from tool support (e.g., maintenance of the Component Folders). This will provide the basis for identifying requirements that quantify the support required. IT will then use these requirements to identify candidate CM tools and to evaluate their ability to support the SN CM process. It is strongly recommended that this requirements-based evaluation be performed before IT procures and deploys any CM tools to support the SN CM process.

Some tools used by previous network CM efforts include:

- Continuus (Continuus)
- ClearCase (Atria)
- SA-EXPERTISE (Software Artistry)
- Microsoft Access

**Establish SN Change Control Board**

The SN Change Control Board is an integral part of the SN CM process -- without the SN CCB, there is no mechanism to control SN architectural changes. Establishment of the SN CCB enables all SN organizations to “buy-in” into the SN CM process and identifies unexpected organizational issues which must be
resolved for the process to work successfully  Since the SN CCB is such a critical component of the SN CM process, the CCB must be in place before transition to this process.

Train SN Personnel

This CM process changes current SN operations, maintenance, and engineering procedures; consequently, all SN personnel should be trained in the new CM process. Since people generally retain information better when they are required to use it, IT should consider conducting the training no more than 30 days prior to transition to the SN CM process.

IT has already tasked development of this training as a part of this task order.

APPENDIX B-1 SAMPLE CM FORMS

<table>
<thead>
<tr>
<th>SN ENGINEERING REQUEST</th>
<th>Control Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Originator: Name:</td>
<td>Phone: Date:</td>
</tr>
<tr>
<td>Organization:</td>
<td></td>
</tr>
<tr>
<td>Incident Description:</td>
<td></td>
</tr>
</tbody>
</table>
Suggested Resolution:

<table>
<thead>
<tr>
<th>Urgency:</th>
<th>High</th>
<th>Medium</th>
<th>Low</th>
<th>Continuation</th>
</tr>
</thead>
</table>

Analyst: Name: Organization: Phone: Date:

Analysis of Incident:

Recommended Resolution:

Resolution: No Change Component Architectural (CR Number )

Description:

Approved By: Date:

**FIGURE A-1. SAMPLE ENGINEERING REQUEST FORM**

**SN CHANGE REQUEST**

<table>
<thead>
<tr>
<th>Originator:</th>
<th>Name:</th>
<th>Phone:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Organization:</td>
<td></td>
<td>Date:</td>
</tr>
</tbody>
</table>

Control Number
**Description of Change:**

**Change Justification:**

**Preliminary Assessment:**

**Urgency:** ![High] ![Medium] ![Low] □ Continuation

<table>
<thead>
<tr>
<th>Impact Assessments</th>
<th>IA Control Number</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Architecture Group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation Group</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment Group</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**CCB Decision:**
- Approved
- Rejected
- Reinvestigate

**Description:**

**CCB Chair:**

---

**FIGURE A-2. SAMPLE CHANGE REQUEST FORM**

---

**SN IMPACT ASSESSMENT**

<table>
<thead>
<tr>
<th>Control Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CR Control Number:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analyst:</td>
</tr>
<tr>
<td>----------</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Technical Analysis:

Page

Schedule Impact:

Page

Cost Impact:

Page

Labor Impact:

Page

Alternatives:

Page

Approved By: |

Date: |

FIGURE A-3. SAMPLE IMPACT ASSESSMENT FORM
APPENDIX B-2 CHARTER TEMPLATES

The following template is derived from the CCB Charter outline presented in *Cultivating Successful Software Development: A Practitioner's View*.

SN Configuration Control Board (CCB) Charter Template

1.0 CCB Purpose
The purpose of the SN Change Control Board is to ensure that the Switched Network changes and related programmatic changes (i.e., consideration of proposed cost and schedule changes) are processed in visible and traceable manner. The CCB is the forum in which SN participants get together to discuss what needs to be done, (2) responsible agents are assigned for performing agreed-upon work, and (3) decisions and assigned actions are recorded.

2.0 CCB Membership
The SN CCB will be comprised of the following representatives from the IT Services Directorate:

- Architecture Group - [TBD – identify representative(s) by name and/or title]
- Operation Group - [TBD – identify representative(s) by name and/or title]
- Deployment Group - [TBD – identify representative(s) by name and/or title]

Representatives from other organizations may be invited on an as-needed basis.

[TBD – identify by name and/or title] will be responsible for documenting the CCB meetings.

3.0 CCB Chairperson
The SN CCB Chair is [TBD – identify by name and/or title]. The SN CCB Chair will manage the meeting in such a manner that input and discussion are encouraged from all attendees. Product and programmatic decision authority rests with the SN CCB Chair and is made a matter of record in the meeting documentation.

4.0 CCB Activities
The SN CCB will meet every **[TBD – specify frequency of CCB meetings, also specify any requirements for meeting notification, quorum of attendees, etc.]**

The SN CCB will perform the following activities: **[TBD – the following activities are presented as examples of the SN CCB activities to be included. This list should not be considered complete until it has been reviewed and approved by the IT organizations participating in the SN CCB]**
- Reviewing and managing proposed changes to the SN architecture.
- Reviewing and managing cross-organizational SN issues
- Reviewing and managing the SN configuration management process
- Recording CCB minutes
- Reviewing, approving, and, if necessary, recording changes to CCB minutes from the previous CCB meeting.

### 5.0 CCB Meeting Documentation

The following information will be recorded for each CCB meeting: **[TBD – the following items are presented as examples of the information to be included in the SN CCB minutes. This list should not be considered complete until it has been reviewed and approved by the IT organizations participating in the SN CCB]**

- Meeting date, time, and duration
- List of attendees and their organizations
- Items discussed
- Existing action items
- New action items
- Decisions made
### Change Log (you can select this and make it HIDDEN)

<table>
<thead>
<tr>
<th>Rev</th>
<th>Date</th>
<th>Who</th>
<th>What</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>