Product Line Configuration Management Policy and Procedures

Author: Richard Wallace
Date: April 21, 2000
Contact: Richard Wallace, Quantum Solutions
richard.wallace@emendo-ex-erratum.org

Record of Changes

Change Number	Date	Brief Description	Entered By
1	4/21/2000	Original Publication	R. Wallace

1 Introduction

So, why have a Configuration Management Policy and Procedure document? Why have any configuration management data at all? Both are needed to prevent the pain of development. So, don't read this document if:

- You just love meeting unrealistic deadlines, and you get a huge sense of satisfaction when delivery dates slip yet again.
- Nothing pleases you more than being abused by angry customers, SOS, or Directors of application groups.
- You revel in uncertainty about whether the final release is complete and whether it incorporates all the modifications the customer asked for.
- You have never had a problem because of other team members' failure to interact or communicate effectively.
- You like to work in the dark, with no idea of what your colleagues are up to or what impact they might have on your work.
- You enjoy lying awake at night, wondering whether the software is right.
- When you start work on a new release, it doesn't bother you at all that you can't be sure which changes were integrated into previous releases.
- You are utterly uninterested in knowing which bugs were fixed in previous releases and which are still outstanding.
- You crave labor-intensive reworking problems that you thought were solved in a prior release.
- You love the way there is little, wrong, or no, documentation and have to figure-out what the product does every time you need to modify it.
- You have never felt frustrated at having to juggle version management, change management and process management, and keep everything in sync.
- You are totally unmoved at the prospect of gaining recognition in your organization for recommending and developing a software product that delivers better quality, more satisfied customers and a healthier bottom line.
- You have absolutely no interest in being at the cutting edge of software engineering.
- You hate re-using code and relish the prospect of being asked to rebuild a software product, or a new version, from scratch.

By adherence to a coherent, well-defined policy and procedure for creation, modification, and support of a product, new features and utilities can be created, integrated and shipped without frustration.

So who is responsible for following and implementing Configuration Management, or simply, CM? In short, it's you. Everyone is involved. Each person that is involved with a product has some rôle or responsibility. This is explained further in section 2.1.

So now you know who is responsible for CM, what is configuration management? It is the control of the construction and content of a product as it passes through the software life cycle. The software life cycle phases are concept/requirements, design, development (construction), testing (unit, work-set, system), installation, operation, maintenance, and retirement.

When is CM done? It is done for the entire life cycle of the product. In the case of multiple products with a common code base, or multi-platform basis, the life cycle of the product has no retirement phase.

Where is the CM data used? The data, which is much more than the source files for the product, are use by everyone associated with the product. Customers use it to acquire the right product version. Support uses it to identify known problems, new problems, current, and future features. Managers use it to track features, problems, and completion status for product releases. Developers use it to control what features are being implemented, what source files belong to work-sets[1], what problems have been corrected, and which work-sets belong to product releases.

How do we produce CM data? The rest of this document is concerned with that. Creation and storage of CM data is a means to an end and not an end in itself. CM data is the first and the last data that is used to identify the

[1] A work-set is the collection of architecture, user documentation, product source files, other data files, etc. that are necessary to implement a product feature or solution to a problem report.

architecture of a product; what features a product release has; what solved problems in a work-set are in a product release; and what known problems are in a product release.

2 <u>Policy</u>

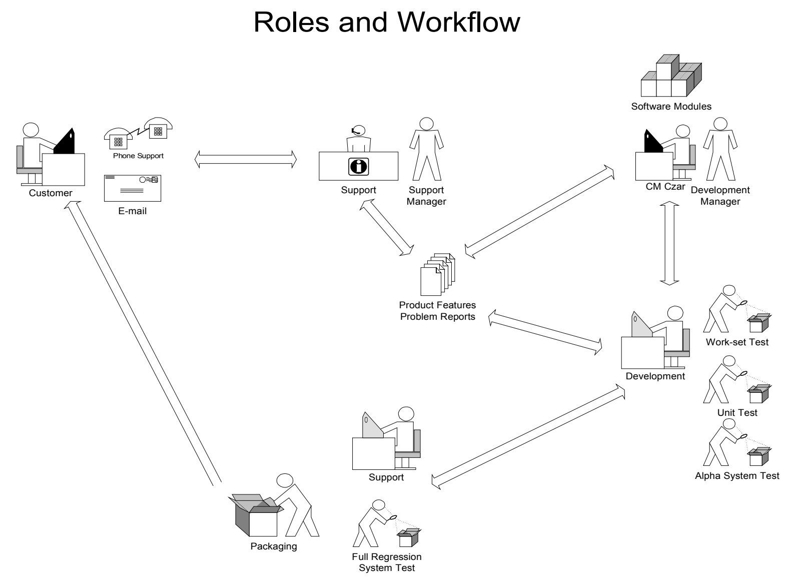

Figure 1 Roles and Workflow

In Figure 1 Roles and Workflow, The policy and procedures for product line configuration management are shown graphically. The following sections detail the rôles and responsibilities.

2.1 Rôles and Responsibilities

2.1.1 Customers

2.1.1.1 Rôle

User of the product

2.1.1.2 Responsibilities

Do:

- Use the product customer documentation.
- Use the public API and utilities for the product.
- Execute information utilities.
- Contact support for configuration setup.
- Report errors correctly.
- Request features.

Do Not:

- Provide diagnostic code testing.

- Have intimate knowledge of the internal processes of the product.

2.1.2 Support Personnel

2.1.2.1 Rôle

First contact with the Customer.

First contact with other support organizations (SOS for example).

Classify incoming problem reports into new or known.

Capture feature requests from customers.

Controls all product releases, both planned and emergency releases.

Sign-off authority for product releases.

2.1.2.2 Responsibilities

Do:

- Follow all CM procedures.
- Identify problem from error identified by product. If the problem is new, create an entry in the problem reporting system.
- Search on-line problem reports for similar problems. Provide known solution if one exits.
- Identify configuration or product API usage errors.
- Answer questions on supported features.
- Co-ordinate problem report solution with CM Czar and Development.
- Update customer documentation if required due to problem report resolution.
- Has authority over and is responsible for all system regression tests[2].
- Package the completed software and documentation for each product release. The completed customer documentation, release notes, product executables/libraries, and customer data files come from the CM data repository. Development places completed software and documentation into the CM data repository.
- Perform all system regression testing.

Do Not:

- Provide general debugging of customers' application.
- Release any software that is not approved by development or support management.

2.1.3 CM Czar

2.1.3.1 Rôle

Central authority for all data in data repositories.

Controlling authority for all data in product releases.

2.1.3.2 Responsibilities

Follow all CM procedures.

Control all CM procedures.

[2] Regression testing is defined as a comprehensive set of system tests that ensure that the product functions correctly. This testing includes success testing and failure testing. Success tests result in correct operation of the product. Failure tests result in known error conditions occurring. Each testing technique results in a known end condition.

Develops and provides mechanisms to generate all reports on data, and data state in the data repositories.

Knows state of all work-set, source files, and product releases.

Tracks all items in the data repository.

Coordinate with Support and Development managers as equal in the CCA.

Monitor adherence to CM procedures.

Has authority over and is responsible for all CM tools, source files, and data repositories.

2.1.4 Support and Development Management

2.1.4.1 Rôle

Controlling authority for product releases

Controlling authority for problem report resolution and feature inclusion in product releases.

2.1.4.2 Responsibilities

Follow all CM procedures.

Control which product features and problem report solutions go into a product release based on completion status and priority.

Create and control personnel assignments.

Create and control product release schedules.

Assign priority for product features and problem report solutions.

Review priorities and completion status of all work-sets.

Recommends approval or disapproval of proposed waivers and deviations from current approved configuration documentation for a configuration controlled item.

Evaluate CM procedures.

2.1.5 Development Personnel

Development personnel are subdivided into project leaders who have purview over a team of individual contributors. Individual contributors report to project leaders. Project leaders report to the development manager.

2.1.5.1 Rôle

Developer of software for product features.

Modifier of software for resolution to problem reports.

Document product features for customer documentation in accordance with defined standards in section 3.

Document problem report solutions for customer documentation in accordance with defined standards in section 3.

2.1.5.2 Responsibilities

Follow all CM procedures.

Calculate labor estimates for personnel assignments.

Coordinate and monitor the completion status of product features and problem report solutions.

Create/Modify product source files to implement product features or problem report solutions.

Define which data files belong to a work-set..

Assess implementation difficulty for product features and problem report solutions.

Has authority over and is responsible for all unit, work-set, and alpha[3] system testing.

Perform all unit, work-set, and alpha system testing.

Development leaders coordinate with support personnel for packaging and distribution of product releases.

2.2 Technical Data

Technical data is recorded information, regardless of the form or method of recording, of a technical nature relating to a product. Technical data is required to define and document an engineering design or product configuration sufficient to allow duplication of the original items and is used to support production, engineering, and customer support activities.

Technical data is part of a work-set package and should include all engineering drawings, associated lists, process descriptions, and customer documentation. This data also includes any other documents that define the physical geometry, composition, performance characteristics, construction, acceptance test procedures, and instructions for the installation, operation, maintenance, and training, including support of a system or equipment.

All technical data shall be maintained in, and issued from, the CM data repository.

2.2.1 Architecture Documentation

2.2.1.1 Creating Features/Utilities

All features and utilities for a product release shall have a Microsoft Word format functional statement recorded in the CM data repository according to the templates in appendix .

Each feature and utility shall have recorded in the CM data repository a Data Flow Diagram (DFD) in Visio format, and an Interface Control Document[4] (ICD) in Microsoft Word format according to the templates in appendix .

Each feature and utility shall have a series of test scenarios descriptions recorded in the data repository. Each scenario shall detail the conceptual steps necessary to initialize, execute, and analyze results to determine successful test execution. Each series of test scenarios should have both success and failure tests.

2.2.1.2 Modifying Features/Utilities

All features and utilities to be modified for a product release shall have recorded in the CM data repository a delta functional statement, DFD, and ICD. If a feature or utility is missing its base-level functional statement, DFD, or ICD, a base-level version of this data shall be created before creation of the delta data.

All features and utilities to be modified for a product release shall have the series of test scenarios modified to reflect the new feature or utility.

2.2.1.3 Deprecating Features/Utilities

All features and utilities to be retired for a product release shall have recorded in the CM data repository a deprecation statement, delta DFD showing the deprecation and a delta ICD showing which interfaces were retired.

All features and utility to be retired for a product release shall have each test scenario marked as retired.

[3] System testing that only tests the functionality of a new product feature or solution to a problem report.

[4] Interface control document depicts physical and functional interfaces of related/co-functioning items

2.2.2 Work-sets

A work-set is comprised of source files, documentation, and tests for a feature or utility belonging to a product. A work-set's contents may be shared between other work-sets. Effort shall be made by development to keep work-set interaction to a minimum without duplication.

2.2.2.1 Source files

Source files consist of, but is not limited to, all programming languages, utility scripts, shell scripting, interface definition, mark-up notations or languages used to implement a product feature or utility.

Effort shall be made to ensure that each source file contains only one function or capability to enhance modularity.

These source files shall be associated with functional statements, DFDs and ICDs from section 2.2.1. These source files and related documentation as described in section 2.2.2.2 shall comprise a work-set.

Each source file shall contain the header and PVCS macros shown in appendix 4.1.

2.2.2.2 Documentation

Work-set documentation shall consist of check-in/check-out change descriptions, detailed functional descriptions for each function in a source file.

Publicly accessible application programming interfaces or utility functions that support the product shall have Customer Documentation as described by section 2.2.3.

2.2.2.3 Tests

Tests shall be added, modified or retired by Development and Support.

Each test or suite of tests, from the following capabilities shall have the following documentation: what feature or capability is being tested, how to setup the test, initial values for the test, execution instructions for the test, and expected result or results of the test.

Unit Testing

Each function or capability shall have at least one unit test that checks correct response, error/exception response, or argument bounds/value bounds.

When a function is retired, its unit tests are retired.

Work-Set Testing

By definition, a work-set is a feature or utility belonging to a product. Each work-set shall have a suite of tests that check for correct feature or utility operation, error/exception responses, arguments or values, and bounds checking for arguments or bounds. Work-Set testing may be a series of unit tests for the work-set.

Individual tests, or test scenarios, shall be written to validate that the new or modified feature or utility does not conflict or corrupt with existing product co-functionality that the feature or utility is being added or modified.

When a feature or utility of the product is retired, its work-set tests are retired.

System Alpha Testing

System alpha testing shall be a suite of end-to-end testing of the product showing that the customer accessible portion of the added or modified product feature or capability is functional. System Alpha tests may be a series of work-set tests for a specific feature or capability of a product version.

When a feature or capability of the product is retired, its system alpha tests are retired.

System Regression Testing

System regression testing shall ensure that the current version of the customer available product is downwardly compatible with prior versions of the product with the exclusion of retired features and utilities. System regression tests may b a series of system alpha tests. All customer accessible features and functions of the product shall be checked for correct response, error/exception response, and argument bounds/value bounds.

2.2.3 Customer Documentation

Development and Support shall be responsible for writing and modifying customer documentation in concert with writing and modifying work-set data. Each guide described below may be combined into one or more physical documents. These documents shall be downloadable by the customer from a well-known web site.

Customer documentation shall be issued in conjunction with product releases such that customer documentation shall be at the same version nomenclature as the product.

2.2.3.1 Configuration and Installation Guide

Each customer accessible configuration option or installation option shall be described. The information provided shall detail the name, purpose, function, and valid values for a configuration and installation.

The guide shall contain complete working examples of correct sequence of operations for each correct product configuration and installation.

2.2.3.2 Programmers' Reference

Each customer accessible API shall be described. The information provided shall detail the name, purpose, data type descriptions, valid input values, return values, modified arguments in the API, and usage instructions for the API. The organization of the document shall be effective as a quick reference. A detailed table of contents and document index shall be created.

2.2.3.3 Programmers' Guide

The guide shall contain complete working programs or references to external source files that are complete fully commented working programs. These external source files shall be considered part of the document and shall be contained in the CM data repository.

2.2.3.4 Utility Usage and Installation Guide

Each customer accessible product utility, and its installation, shall be described. The information provided shall detail the name, purpose, function, valid input values, and output values for each utility that supports the product. The output from each utility shall be fully described such that a novice customer would be able to understand the output of the utility.

2.2.4 Feature Request/Problem Reporting

2.2.4.1 From Test

All feature requests and problem reports from unit, work-set, alpha system, and system regression testing shall be entered into the issue tracking system. New features work items and problem report resolution work items can be self-assigned (may be reassigned by management), assigned by CCA, or assigned by Development management to responsible individuals.

2.2.4.2 From Customer

All feature requests and problem reports from customer contact shall be entered into the issue tracking system. The CCA or Development management shall assign work items.

3 Procedures

Details of the logical and physical actions, the templates, and the electronic forms that shall be used to create and manage the technical data described in section 2.2 are covered in this section.

In general the following procedures for product feature creation or problem resolution are followed:

1. Management creates work item and assigns personnel to the work item.
2. Assigned personnel associate the work item with all technical data from section 2.2. The associated work item and technical data now becomes a work-set. Management notified by CM when a work-set is created. Management modifies work item status.
3. Check-out of technical data from the CM repository by assigned personnel.
4. Create, or modify, the technical data by assigned personnel.
5. Perform walk-through/peer check of technical data. If unsuccessful, iterate to step 3.
6. Assigned personnel perform unit and work-set tests. If unsuccessful, iterate to step 3.
7. Check-in the technical data to the CM repository.
8. Management views work-set status and modifies work item status for work-set.
9. Check-out the technical data from the CM repository by assigned personnel.
10. Assigned personnel perform alpha system testing. If unsuccessful, modify technical data and iterate to step 3.
11. Check-in the technical data to the CM repository.
12. Management views work-set status and modifies work item status for work-set.
13. Support checks-out the technical data from the CM repository and performs regression system testing. If unsuccessful, iterate to step 3.
14. Management approves all work-sets for inclusion in a product release.
15. Support creates product release version for technical data in CM repository.
16. Support creates installation kit from the product release version in the CM repository.
17. Support distributes product release to customers.

These steps are shown in process flow form in Figure 2 General Process Flow.

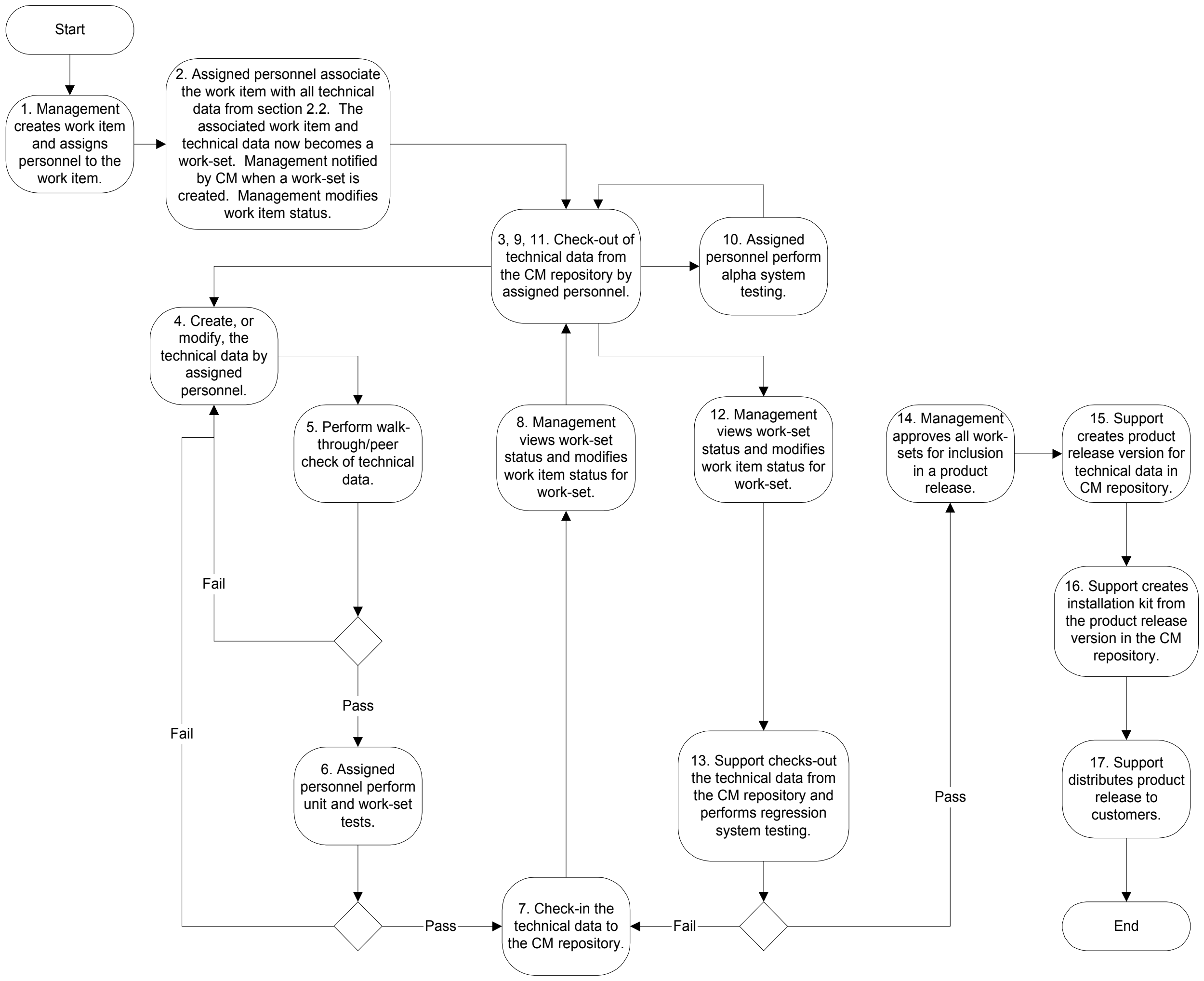
Start
1. Management creates work item and assigns personnel to the work item.
2. Assigned personnel associate the work item with all technical data from section 2.2. The associated work item and technical data now becomes a work-set. Management notified by CM when a work-set is created. Management modifies work item status.
3, 9, 11. Check-out of technical data from the CM repository by assigned personnel.
10. Assigned personnel perform alpha system testing.
4. Create, or modify, the technical data by assigned personnel.
5. Perform walk-through/peer check of technical data.
Fail
Pass
Fail
6. Assigned personnel perform unit and work-set tests.
Pass
7. Check-in the technical data to the CM repository.
8. Management views work-set status and modifies work item status for work-set.
12. Management views work-set status and modifies work item status for work-set.
13. Support checks-out the technical data from the CM repository and performs regression system testing.
Fail
Pass
14. Management approves all work-sets for inclusion in a product release.
15. Support creates product release version for technical data in CM repository.
16. Support creates installation kit from the product release version in the CM repository.
17. Support distributes product release to customers.
End

Figure 2 General Process Flow

3.1 Architecture documents

Architecture documents are the first documentation for a product feature or utility. These documents are the last documents finished for a product feature or utility. This is due to the importance of these documents to describe what the product does. Simply "reading the code" does nothing to enhance understanding for why a product or utility operates in a certain way. Correctly-constructed DFDs and product specifications that are linked together form a cohesive "picture" of a product's architecture. While it is desirable to have a single, overall architecture document for a product it is not absolutely necessary. Tightly integrated product feature and utility architecture documents can replace a single architecture document.

3.1.1 Format

In the case of creating a feature/utility as described in section 2.2.1.1 each of the functional statements for the work items shall be referenced in an architectural document following the template as shown in .

3.1.2 Control and Review

Architecture documents shall be source controlled as any other technical data with specific revisions created, peer reviewed and maintained in the CM repository. Each publication of the document shall have a clear progressive version label. No two versions of the document shall have the same version label.

As stated in section 2.2.1 documents shall be in Microsoft Word format. Comments and edits to documents shall use the Microsoft Word product feature to highlight changes. These edits shall be preserved as incremental checked-in data before the final edits being made and the document made available use.

The process flow for document creation is shown in section 4.2, Figure 5 Document Review Process Flow.

3.1.3 Peer Review

The architectural peer review for a work-set is presented to team, and support individuals for their comments and approval. The focus of the first review is the DFD and ICD technical data.

The review follows the process shown in section 4.3.

3.1.4 Approval

Management shall direct the editor and CM Czar to "freeze" a particular version for final publication via web-site. This frozen version shall be the approved architecture document for use by management and team.

3.2 User Documents

The term "User" has multiple meanings based on who the "user" is. For Product Line Configuration Management the users are:

- Development
- Support
- Customer

3.2.1 Development

3.2.1.1 Format

Development personnel shall create detail design documents for the elements in the architecture following template as shown in .

3.2.1.2 Control and Review

Detail design documents shall be source controlled as any other technical data with specific revisions created, peer reviewed and maintained in the CM repository. Each publication of the document shall have a clear progressive version label. No two versions of the document shall have the same version label.

As stated in section 2.2.1 documents shall be in Microsoft Word format. Comments and edits to documents shall use the Microsoft Word product feature to highlight changes. These edits shall be preserved as incremental checked-in data before the final edits being made and the document made available use.

The process flow for document creation is shown in section 4.2, Figure 5 Document Review Process Flow.

3.2.1.3 Peer Review

The detailed design peer review for a work-set is presented to team, and support individuals for their comments and approval. The focus of the second review is the mapping of function and utilities to the DFDs and ICDs for the work item.

The review follows the process shown in section 4.3.

3.2.1.4 Approval

Management shall direct the editor and CM Czar to "freeze" a particular version for final publication via web-site. This frozen version shall be the approved detailed design document for use by management and team.

3.2.2 Test Procedures

3.2.2.1 Format

Development personnel shall create test documents for the functions and utilities detailed in the detailed design following template as shown in . This format shall be used for unit, work-set, alpha system, and regression system, testing. The content of the documents should be aggregated starting from unit testing progressing through work-set testing to alpha system testing. The aggregation is show graphically in Figure 3 Test Aggregation.

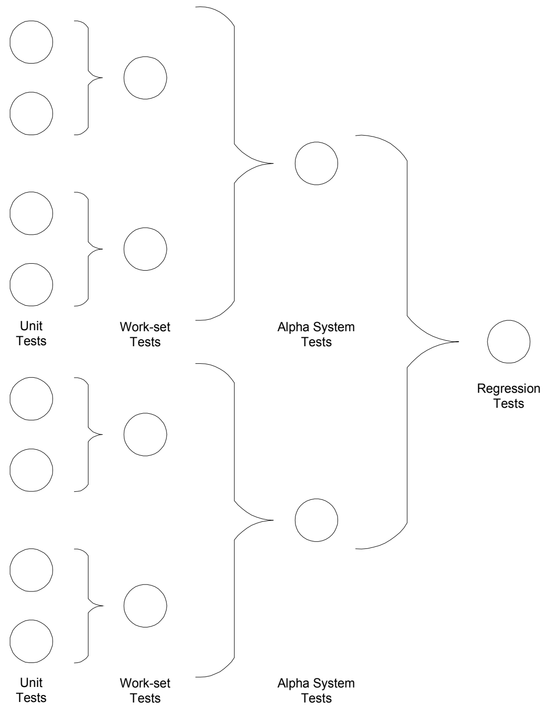

Figure 3 Test Aggregation

The aggregation shows how each low-level unit test is used to build work-set tests onward to the product regression test suite.

3.2.2.2 Control and Review

Test documents shall be source controlled as any other technical data with specific revisions created and maintained in the CM repository. Each publication of the document shall have a clear progressive version label. No two versions of the document shall have the same version label.

As stated in section 2.2.1 documents shall be in Microsoft Word format. Comments and edits to documents shall use the Microsoft Word product feature to highlight changes. These edits shall be preserved as incremental checked-in data before the final edits being made and the document made available use.

The documents shall be used as the controlling description of a test or test suite. Test sources are coded to perform as written in the test documents.

3.2.3 Support

3.2.3.1 Format

Support personnel shall create test documents to regression test use scenarios of product functions and utilities as detailed in the detailed design following template as shown in .

3.2.3.2 Control and Review

Test documents shall be source controlled as any other technical data with specific revisions created and maintained in the CM repository. Each publication of the document shall have a clear progressive version label. No two versions of the document shall have the same version label.

As stated in section 2.2.1 documents shall be in Microsoft Word format. Comments and edits to documents shall use the Microsoft Word product feature to highlight changes. These edits shall be preserved as incremental checked-in data before the final edits being made and the document made available use.

The documents shall be used as the controlling description of a test or test suite. Test sources are coded to perform as written in the test documents.

3.2.3.3 Peer Review

The peer review for system regression testing is presented to team, and support, and management individuals for their comments and approval. The focus of the review is assurance that approved and supported configurations of the product are tested.

The review follows the process shown in section 4.3.

3.2.4 Customer

3.2.4.1 Format

Support and development personnel shall create the documents detailed in section 2.2.3.

3.2.4.2 Control and Review

Customer documents shall be source controlled as any other technical data with specific revisions created and maintained in the CM repository. Each publication of the document shall have a clear progressive version label. No two versions of the document shall have the same version label.

As stated in section 2.2.1 documents shall be in Microsoft Word format. Comments and edits to documents shall use the Microsoft Word product feature to highlight changes. These edits shall be preserved as incremental checked-in data before the final edits being made and the document made available use.

3.2.4.3 Peer Review

The peer review for customer documentation is presented to team, and support, and management individuals for their comments and approval. The focus of the review is accuracy, completeness, and clarity of the user documentation for personnel not intimate with the internal operation of the product features and utilities.

The review follows the process shown in section 4.3.

3.2.4.4 Approval

Management shall direct the editor and CM Czar to "freeze" a particular version for publication via web-site for the customer product documentation. The publicly available customer documents shall be current for the publicly available product. This frozen version shall be the approved user documentation use by customers, management, and development and support teams.

3.3 Problem Reports/Feature Requests

The function of problem reports and feature requests is to support each step of the system change request (SCR) cycle. The SCR is used to track both problem reports and feature requests. Each SCR is assigned a unique tracking number.

Table 1 SCR States shows the life-cycle states an SCR.

NAME	DEFINITION
Open	A new SCR.
Abeyant	In a state of waiting before a transition back to Open. An abeyant SCR is one for which further progress toward a resolution cannot currently be made. Any SCR in this state should have description text identifying what action will cause the SCR to come out of abeyance. Abeyant SCRs must be resolved as part of the test exit criteria.
Accepted	The SCR has sufficient data to proceed with work.
Deferred	A disposition resulting from joint agreement of the assigned individual working the SCR and management that the SCR does not represent a problem or concern for the current release. SCR text should indicate when the SCR should be re-evaluated and transition to the abeyant state. (e.g., deferred until release 2.x.)
Rejected	A disposition resulting from joint agreement of the assigned individual working the SCR, the originator, and product management that the SCR does not represent a problem or concern for the project, or the described problem has been covered in a previous SCR. SCR text should include the reason why it was rejected.
Fixed	A disposition resulting from a resolution to the SCR. This fix may be the result of a code modification, configuration change, documentation update, or process change.
Verified	The support personnel will verify a SCR if the SCR was originally opened by a customer. If development or support the SCR this step will be skipped.
Reopened	A problem that was reported Fixed still exists. The problem was not corrected or reappeared through regression testing. SCR text should include any retest results that provide new or more information.
Closed	A disposition resulting from a transition through Fixed, and Verified for customer SCRs.

Table 1 SCR States

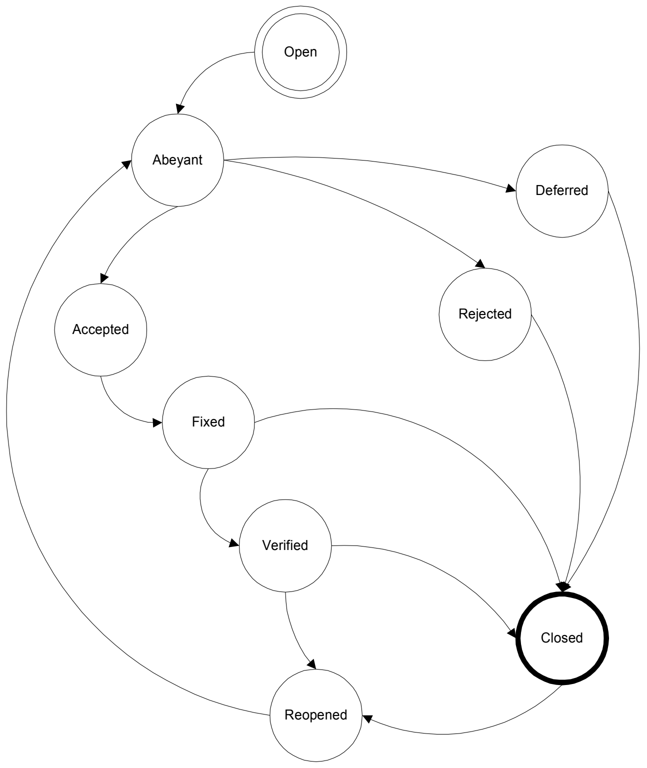

Figure 4 SCR State Transitions

Table 2 SCR Severity Levels shows the different levels of severity of an SCR.

Severity	Criticality	Description
1	Show-Stopper	Testing has stopped early in the test process and cannot continue without a fix. A critical problem can also occur later in the test cycle due to a serious regression.
2	Critical	Testing is severely limited, usually caused by an inoperative or undelivered piece of major functionality. Testing on other major functions can continue.
3	Functionality	A problem with the functionality was detected that is of importance to a production implementation. It is not a 'show-stopper', but has an impact to the overall quality of the release.
4	Minor	Generally a 'nit', a very minimal problem having little or no impact on production with the current release.
5	Enhancement	A suggested improvement to the product, e.g., usability, performance, or expanded functionality.

Table 2 SCR Severity Levels

Each controlling user (product development, support, and management) of the SCR system shall have a unique name, access control list, and a password for administrator accounts. All personnel shall use the SCR system as described in section .

Customers shall have a restricted access to the SCR via a web-site to view all SCRs for the current publicly available product.

The ability to track SCRs through a project is important to all personnel involved with a project. The customers need assurances that the product is stable and has all requested functionality. Management needs to ensure all problems have been resolved before deployment and the development team needs to see what problems have been found, so resolutions can be implemented.

Once a SCR is opened, the SCR is generally processed through numerous steps before it is fully resolved and closed. Placing a SCR into a "Fixed" indicates that the correction has been made to the affected source module or document directly.

3.4 Build Procedures

All software shall be controlled by use of the Mernat PVCS Configuration Builder (PVCS-CB) for software builds. Software builds shall follow the following process:

All builds shall be done by version label and shall be automated by use of PVCS-CB.

There shall be four types of builds:

- Ad hoc builds -- These builds occur at any time by an individual checking-out a version label of software to build all or part of the product for unit testing.
- Incremental builds -- These builds shall occur on Tuesdays and Thursdays. It is the controlled product builds for work-set testing.
- Weekly builds -- These builds shall occur on Sundays. It is the controlled product builds for alpha system testing. If Incremental Builds.
- Base Level Builds -- These builds occur on an as needed basis for final product build for regression testing.

Unapproved code shall not be migrated into the source base for Weekly or Base Level builds.

4 Formats and Processes

4.1 Source file Comment Header Format

The following header is required for each source file that is inserted into the CM repository. The syntax used here is for a C/C++ file. Appropriate comment characters shall be used for each supported language. The text between the "<" and ">" is self-explanatory.

The "`$Log$`" is a PVCS macro that the version manager uses. This is required only once in a source file in the top-most comment block.

It is required that this comment header minus the "`$Log$`" PVCS macro be used for every function in the source file.

```
/*
*******************************************************************************
**
** <Function/Module Name>
**
** Description:
** <Clear and complete functional description of what the function/module does.
** Brevity is not wanted here>
**
** $Log$
*/
```

4.2 Document Review

As determined by management assignment, a single person shall act as editor for the document. The process flow for document creation is shown in Figure 5.

1. First draft of document created. Editor assigned by management.
2. Editor checks-in document to CM repository.
3. Editor checks-out document. Peer critique of draft document.
4. Critique from all peers preserved by editor and checked-in to CM repository.
5. Next draft of document created incorporating critique. Iterate to step 4.2.
6. Final draft of document approved by management and team.
7. Document published via web-site and followed by management and team.
8. Document changes requested for additions/revisions due to new features or changes. Iterate to step 4.2.

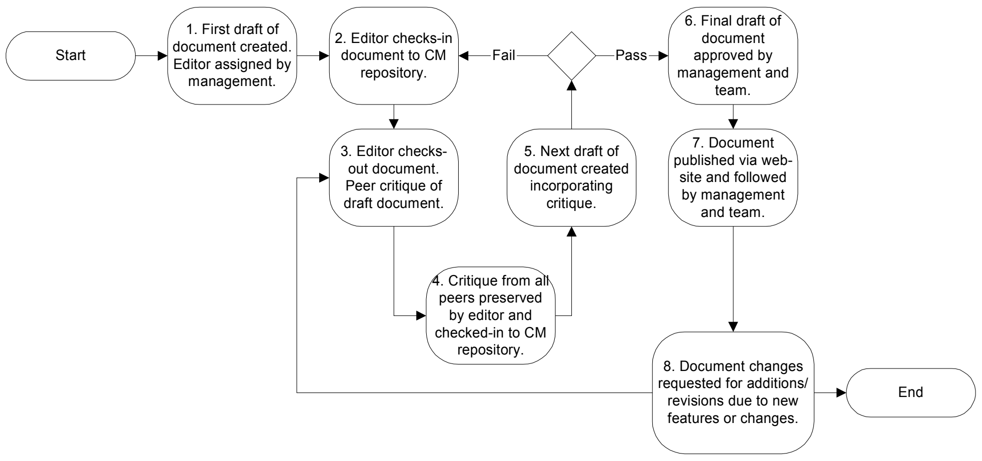
Start
1. First draft of document created. Editor assigned by management.
2. Editor checks-in document to CM repository.
Fail
Pass
6. Final draft of document approved by management and team.
3. Editor checks-out document. Peer critique of draft document.
5. Next draft of document created incorporating critique.
7. Document published via web-site and followed by management and team.
4. Critique from all peers preserved by editor and checked-in to CM repository.
8. Document changes requested for additions/ revisions due to new features or changes.
End

Figure 5 Document Review Process Flow

4.3 Peer Review Process

The peer review process is consistent for each type of peer step 3 of Figure 2 General Process Flow. Reviews typically occur three times during the progress on a work item.

- The first review is focused on the DFD and ICD design.
- The second review focus — nominally at the mid-point of the scheduled time for the work-item — is the mapping of functions and utilities to the DFDs and ICDs for the work item.
- The third review focus is on the adherence of the as-built work item to the build-to technical data. It is the exit review before system testing.

The time required to prepare and execute the reviews should be no more than 16 man-hours for each review. Applicable personnel have to be involved in each review. The review may or may not involve all personnel on the team.

The review process follows. It is shown graphically shown in Figure 6.

1. At least 48 hours prior: The presenter holding the review prepares materials. Appropriate materials for the type of review are placed in a public location for downloading by reviewers. This material is checked-out of the CM Repository.
2. At least 48 hours prior: Presenter schedules room and sends e-mail to applicable personnel.

3. At least 24 hours prior: Reviewers acquire review materials from the public location.
4. At least 24 hours prior: Reviewers read and comment on materials. These comments are brought to the review.
5. At review: Presenter leads review. Reviewers present comments in line with the agenda of the Presenter. Reviews shall last no longer than 4 hours. Reviews should be targeted to last 2 hours or less.
6. At review: Determine if a second review of the same material is necessary. A second review is only necessary if the presenter and reviewers fail to find the material acceptable even after the comments, amendments, and changes are made to the review materials.
7. After review: Presenter incorporates comments, amendments, and changes, etc. as applicable and places the modified materials in the public location. If a second review is necessary, iterate to step 4.3, if not check-in the material to the CM Repository.

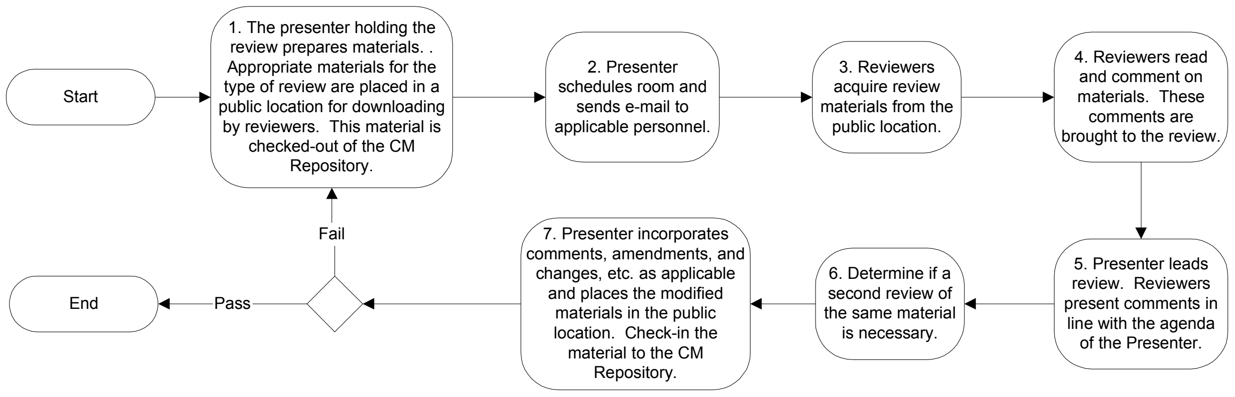
Start
1. The presenter holding the review prepares materials. . Appropriate materials for the type of review are placed in a public location for downloading by reviewers. This material is checked-out of the CM Repository.
2. Presenter schedules room and sends e-mail to applicable personnel.
3. Reviewers acquire review materials from the public location.
4. Reviewers read and comment on materials. These comments are brought to the review.
5. Presenter leads review. Reviewers present comments in line with the agenda of the Presenter.
6. Determine if a second review of the same material is necessary.
7. Presenter incorporates comments, amendments, and changes, etc. as applicable and places the modified materials in the public location. Check-in the material to the CM Repository.
Fail
Pass
End

Figure 6 Peer Review Process Flow