US Department of Transportation
Federal Railroad Administration

Introduction to Software Configuration Management

Office of Safety
Washington, DC 20590

8/8/2006 Version 1.1

FOREWORD

This document provides an introductory overview of the topic of software configuration management (SCM). It is intended to provide FRA headquarters, field, and industry personnel a basic understanding of the topic as well as general guidelines pertaining to the Software Management Control Plan (SMCP) as specified in Title 49 Code of Federal Regulations (CFR) § 236.18. It provides a general outline of the type of information that FRA would expect to be found in an SMCP. The applicable Federal regulation requires that railroads develop, adopt, and implement an SMCP appropriate to their needs and consistent with their business case. The general guidelines in this document must be supplemented by good judgment in handling individual situations.

Submit any questions or requests for clarifications regarding the content of this document to the Staff Director, Signal and Train Control Division, Federal Railroad Administration, RRS-13, Mail Stop 25, 1120 Vermont Avenue, NW, Washington, DC 20005, Fax (202) 493-6216.

TABLE OF REVISIONS

Revision	Date	Change
1.0	5/12/2006	Initial Release
1.1	8/8/2006	Errata- Correction of Numbering in Section 3.3 "Sample Plan Outline"

TABLE OF CONTENTS

1. Introduction

Title 49 CFR § 236.18, "Software Management Control Plan", requires that railroads develop and adopt within 6 months of the effective date of the rule, and fully implement within 30 more months, a software management control plan (SMCP) for their processor-based signal and train control systems. An SMCP is a plan designed to ensure that the proper and correct version of software for each specific site and location on the railroad is documented and maintained throughout the life-cycle of the system. The plan further describes how the proper software configuration is to be identified and confirmed in the event of replacement, modification, or disarrangement of any part of the system.

1.1. Purpose

This document is intended to introduce the topic of software configuration management to personnel who may have never previously been involved with that subject or process. It provides guidance on various considerations for developing, adopting, and implementing an SMCP. It is intended for use or reference by FRA HQ and field personnel, as well as railroad or vendor personnel.

FEDERAL REGULATION REQUIRES THAT RAILROADS SUBJECT TO 49 CFR SECTION 236.18 DEVELOP, ADOPT, AND IMPLEMENT A SOFTWARE MANGEMENT CONTROL PLAN APPRORIATE TO THEIR BUISNESS CASE.

Developing, adopting, and implementing an SMCP is the responsibility of the railroad. The railroad, or a vendor for the railroad, may prepare it. FRA will not mandate any particular format nor actually approve the plan. FRA will however monitor the railroads development, adoption, and implementation of a plan, and will evaluate if the railroads plan fulfills the requirements of § 236.18(c). FRA will then monitor the railroads on-going compliance to their plan. Since the process and procedures are specified by the railroads themselves in their SMCP to best support their business model while meeting the objectives specified § 236.18(c), the expectation is that FRA will find that the railroads will achieve and maintain compliance with those processes and procedures. Situations of railroad non-compliance will be addressed as possible violations of § 236.18.

The Associate Administrator for Safety, or his or her designee, will have final authority to decide the suitability of an SMCP to support the requirements of § 236.18(c). Each SMCP will vary, depending on the individual railroad and property concerned. While the following guidelines have been written taking this into account, they should NOT be considered a substitute for good judgment, experience, and common sense.

1.2. Distribution

This document should be distributed to:

- All personnel in the Office of Safety Assurance and Compliance, Signal and Train Control Division (S&TC), Washington headquarters;

- Division Chiefs, Office of Safety Assurance and Compliance, Washington headquarters;
- Regional Administrators, Deputy Regional Administrators, Regional S&TC Supervisory Specialists, and Federal and State S&TC Inspectors;
- Office of Railroad Research and Development; and,
- And, it should be made available to interested railroad personnel, product suppliers, and/or members of the general public.

1.3. Definitions

The terms and definitions listed below are compiled as an aid to understanding and applying the SCM principles and processes used to manage software development, testing, and maintenance efforts.

a. **ALLOCATED BASELINE (ABL)** - The initially approved documentation that describes an item's functional, interoperability, and interface characteristics that are allocated from those of a system or a higher level configuration item, interface requirements with interfacing configuration items, additional design constraints, and the verification required to demonstrate the achievement of those specified characteristics.

b. **ALLOCATED CONFIGURATION DOCUMENTATION (ACD)** - The approved Allocated Baseline plus approved changes.

c. **AS-BUILT** - Defines the initial software, hardware, or system configuration as it actually has been built.

d. **AUDIT** - An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or criteria.

e. **BASELINE** - A configuration identification document or set of such documents formally designated and fixed at a specific time during the configuration item's life-cycle. Baselines, plus approved changes from those baselines, constitute the current configuration identification.

f. **CHANGE REQUEST (CR) FORM** - A vehicle used to report deficiencies or enhancements generated against CIs or technical data; a document that requests a correction or change to the baseline documentation and software.

g. **COMPUTER SOFTWARE (or SOFTWARE)** - A combination of associated computer instructions and computer data definitions required to enable the computer hardware to perform computational or control functions.

h. **COMPUTER SOFTWARE CONFIGURATION ITEM (CSCI)** - A configuration item that is software.

i. **CONFIGURATION** - The functional and physical characteristics of existing or planned hardware, firmware, or software or a combination thereof, as set forth in technical documentation and achieved in a product.

j. **CONFIGURATION AUDIT** - A formal examination of a CSCI. Two types of configuration audits exist: the Functional Configuration Audit (FCA) and the Physical Configuration Audit (PCA).

k. **CONFIGURATION CONTROL** - The systematic proposal, justification, evaluation, coordination, and approval or disapproval of proposed changes, and the implementation of all approved changes in the configuration of a CI after establishment of the baseline(s) for the CI.

l. **CONFIGURATION IDENTIFICATION** - The selection of CIs; the determination of the types of configuration documentation required for each CI; the issuance of numbers and other identifiers affixed to the CIs and to the technical documentation that defines the CI's configuration, including internal and external interfaces; the release of CIs and their associated configuration documentation; and the establishment of configuration baselines for CIs.

m. **CONFIGURATION ITEM (CI)** - An aggregation of hardware or software that satisfies an end use function and is designated for separate configuration management.

n. **CONFIGURATION STATUS ACCOUNTING (CSA)** - The recording and reporting of information needed to manage CIs effectively, including:

   (1) A record of the approved configuration documentation and identification numbers.

   (2) The status of proposed changes, deviations, and waivers to the configuration.

   (3) The implementation status of approved changes.

   (4) The configuration of all units of the CI in the operational inventory.

o. **DELIVERABLE** - A system or component that is obligated contractually to a customer or intended user.

p. **DEVELOPMENTAL CONFIGURATION** - The software and associated technical documentation that define the evolving configuration of a CSCI during development. It is under the development contractor's or procuring organization's configuration control and describes the software design and implementation. The Developmental Configuration may be stored on electronic media.

q. **DEVIATION** - A specific written authorization to depart from a particular requirement(s) of an item's current approved configuration documentation for a specific number of units or a specified period of time.

r. **ENGINEERING CHANGE PROPOSAL (ECP)** - A proposed engineering change and the documentation by which the change is described, justified, and submitted to the for approval or disapproval.

s. **FIRMWARE** - The combination of a hardware device and computer instructions or computer data that reside as read-only software on the hardware device. The software cannot be readily modified under program control.

t. **FUNCTIONAL BASELINE (FBL)** - The initially approved documentation describing a system's or item's functional, interoperability, and interface characteristics and the verification required to demonstrate the achievement of those specified characteristics.

u. **FUNCTIONAL CONFIGURATION AUDIT (FCA)** - The formal examination of functional characteristics of a CI, prior to acceptance, to verify that the CI has achieved the requirements specified in its functional and allocated configuration documentation.

v. **FUNCTIONAL CONFIGURATION DOCUMENTATION (FCD)** - The approved FBL plus approved change.

w. **NONDEVELOPMENTAL SOFTWARE (NDS)** - Deliverable software that is not developed under the contract but is provided by the contractor, the procuring organization, or a third party. NDS may be referred to as reusable software, government-furnished software, or commercially available software, depending on its source.

x. **NOTICE OF REVISION (NOR)** - A document used to define revisions to drawings, associated lists, or other referenced documents which require revision after ECP approval.

y. **PHYSICAL CONFIGURATION AUDIT (PCA)** - The formal examination of the "as-built" configuration of a CI against its technical documentation to establish or verify the CI's product baseline.

z. **PRODUCT BASELINE (PBL)** - The initially approved documentation describing all of the necessary functional and physical characteristics of the CI and the selected functional and physical characteristics designated for production acceptance testing and tests necessary for support of the CI.

aa. **PRODUCT CONFIGURATION DOCUMENTATION (PCD)** - The approved product baseline plus approved changes.

bb. **PROGRAM MANAGEMENT** - The organization sponsoring the field activity project office.

cc. **PROJECT MANAGEMENT** - The designated organization from the field activity project office responsible for the overall management of specific projects.

dd. **RELEASE** - A configuration management action whereby a particular version of software is made available for a specific purpose (e.g., released to test).

ee. **REUSABLE SOFTWARE** - Software developed in response to the requirements for one application that can be used, in whole or in part, to satisfy the requirements for another application.

ff. **RESOURCES** - The totality of computer hardware, software, personnel, documentation, supplies, and services applied to a given effort.

gg. **SOFTWARE** - See **Computer Software**.

hh. **SOFTWARE CONFIGURATION MANAGEMENT (SCM)** - A discipline that applies technical and administrative direction and surveillance to perform the functions listed below.

(1) Identify and document the functional and physical characteristics of CSCIs.

(2) Control the changes to CSCIs and their related documentation.

(3) Record and report information needed to manage CSCIs effectively, including the status of proposed changes and the implementation status of approved changes.

(4) Audit the CSCIs to verify conformance to specifications, interface control documents, and other contract requirements.

ii. **SOFTWARE DEVELOPMENT LIBRARY (SDL)** - A controlled collection of software, documentation, and other intermediate and final software development products, and associated tools and procedures used to facilitate the orderly development and subsequent support of software. The SDL includes the Developmental Configuration as part of its contents. The SDL provides storage of and controlled access to software development products in human-readable form, machine-readable form, or both. This library may also contain management data pertinent to the software development project.

jj. **SOFTWARE-RELATED GROUP** - Project members responsible for generating requirements, design, development, validation, verification, documentation, maintenance, and logistics of software.

kk. **SOFTWARE SUPPORT** - The sum of all activities that take place to ensure that implemented and fielded software continues to fully support the operational mission of the software.

ll. **SOFTWARE UNIT** - An element in the design of a software item; for example, a major subdivision of a software item, a component of that subdivision, a class, object, module, function, routine, or database. Software units may occur at different levels of a hierarchy and may consist of other software units. Software units in the design may or may not have a one-to-one relationship with the code and data entities (routines, procedures, databases, data files, etc.) that implement them or with the computer files containing those entities.

mm. **SOFTWARE TEST ENVIRONMENT** - A set of automated tools, firmware devices, and hardware necessary to test software. The automated tools may include but are not limited to test tools such as simulation software, code analyzers, test case generators, path analyzers, etc., and may also include the tools used in the software engineering environment.

nn. **SPECIFICATION CHANGE NOTICE (SCN)** - A document used to propose, transmit, and record changes to a specification.

oo. **TECHNICAL REVIEW** - An activity by which the technical progress of a project is assessed relative to its technical or contractual requirements. The review is conducted at logical transition points in the development effort to identify and correct problems resulting from the work completed thus far before the problems can disrupt or delay the technical progress. The review provides a method to determine that the development of a CSCI and its documentation, have met contract requirements.

pp. **VERSION** - An identified and documented body of software. Modifications to a version of software (resulting in a new version) require configuration management actions, by the contractor, the procuring organization, or both.

qq. **WAIVER** - A written authorization to accept an item, which during manufacture, or after having been submitted for inspection or acceptance, is found to depart from specified requirements, but nevertheless is considered suitable for use "as is" or after repair by an approved method.

1.4. References

MIL-STD-498	Software Development and Documentation, 5 December 1994
MIL-STD-973	Configuration Management, 17 April 1992
MIL-M-38784C	Military Specification - Manuals, Technical: General Style and Format Requirements, 12 October 1990
NAVAIRINST 4130.1C	Naval Air Systems Command Configuration Management Policy, 31 January 1992
IEEE-STD-610-1990	Glossary of Software Engineering Terminology
IEEE STD 828-1990	IEEE Standard for Software Configuration Management Plans
IEEE-STD-1042-1987	IEEE Guide to Software Configuration Management,
CMU/SEI-93-TR-25	Key Practices of the Capability Maturity Model, Version 1.1, February 1993

1.5. Acronyms and Abbreviations

ABL	Allocated Baseline
ACD	Allocated Configuration Documentation
AM	Acquisition Manager
CAD	Computer-Aided Design
CALS	Continuous Acquisition and Life-Cycle Support
CAM	Computer-Aided Manufacturing
CCB	Configuration Control Board
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CI	Configuration Item
CITIS	Contractor Integrated Technical Information Service

CM	Configuration Management
CMU	Carnegie Mellon University
COM	Computer Operation Manual
COTS	Commercial Off-The-Shelf
CPM	Computer Programming Manuals
CSA	Configuration Status Accounting
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
DBDD	Database Design Description
DID	Data Item Description
DM	Data Management
DoD	Department of Defense
DTP	Desktop Procedure
ECP	Engineering Change Proposal
EM	Engineering Master
FBL	Functional Baseline
FCA	Functional Configuration Audit
FCD	Functional Configuration Documentation
FPC	Functional and Physical Characteristics
FQT	Functional Qualification Testing
FSM	Firmware Support Manuals
HWCI	Hardware Configuration Item
ICWG	Interface Control Working Group
ID	Identification

IDD	Interface Design Document
IRS	Interface Requirements Specification
MCCR	Mission Critical Computer Resources
NDS	Non-Developmental Software
NOR	Notice of Revision
OCD	Operational Concept Description
OT&E	Operational Testing and Evaluation
PBL	Product Baseline
PCA	Physical Configuration Audit
PCD	Product Configuration Documentation
PDR	Preliminary Design Review
PM	Program Manager
QA	Quality Assurance
SCCB	Software Configuration Control Board
SCM	Software Configuration Management
SCMP	Software Configuration Management Plan
SCN	Specification Change Notice
SCOM	Software Center Operator Manual
SCP	Software Change Proposal
SCR	Software Change Request
SCRB	Software Change Review Board
SDD	Software Design Document
SDF	Software Development File
SDL	Software Development Library
SDP	Software Development Plan

SDR	System Design Review
SEI	Software Engineering Institute
SEP	Software Enhancement Proposal
SIOM	Software Input/Output Manual
SIP	Software Installation Plan
SPS	Software Product Specification
SQA	Software Quality Assurance
SRR	Software Requirements Review
SRS	Software Requirements Specification
SSA	Software Support Activity
SSDD	System/Segment Design Document
SSR	Software Specification Review
SSS	System/Subsystem Specification
STD	Standard
STP	Software Test Plan
STR	Software Test Report
STR Form	System Trouble Report Form
STrP	Software Transition Plan
SUM	Software User's Manual
SVD	Software Version Description
TRR	Test Readiness Review
V&V	Verification and Validation
VDD	Version Description Document

2. Software Configuration Management

Software Configuration Management (SCM) is defined as a process for establishing and maintaining consistency of a software product's performance, and functional and physical attributes with its requirements, design, and operational information throughout its life. The SCM process is embodied in rules, procedures, techniques, methodology, and resources to assure that:

- The configurations of the system and/or item (its attributes) are documented.
- Changes made to the item in the course of development, production, and operation, are beneficial and are effected without adverse consequences.
- Changes are managed until incorporated in all items affected.

The objectives of SCM are summarized as follows:

- Ensure the orderly release and implementation of new/revised software and related documentation.
- Implement only approved changes to both new and existing software.
- Verify that software changes comply with approved specifications.
- Reflect changes and updates in project documentation.
- Provide visibility of changes to project management.
- Evaluate and communicate the impact of changes.
- Prevent unauthorized changes from being made.

2.1. Software Configuration Identification

Software Configuration Identification (SCI) incrementally establishes and maintains the definitive current basis for control and status accounting of a system and its computer software configuration items (CSCIs) throughout their life-cycle (development, production, deployment, and operational support, disposal). Effective SCI is a prerequisite for the other SCM activities (configuration control, status accounting, audit), which all use the products of configuration identification. If CSCIs and their associated configuration documentation are not properly identified, it is impossible to control the changes to the items' configuration, to establish accurate records and reports, or to validate the configuration through audit. Inaccurate or incomplete configuration documentation may result in defective products, schedule delays, and higher maintenance costs after delivery. Good configuration control procedures assure the continuous integrity of the configuration identification. The configuration identification process includes:

- Selecting configuration items;
- Determining the types of configuration documentation required;
- Determining the appropriate configuration control;

- Issuing identifiers for the CSCIs;
- Maintaining the configuration identification of CSCIs;
- Releasing configuration documentation; and,
- Establishing configuration baselines for the configuration control.

There is little consistency in item identification practices among COTS producers, and often little consistency between two products provided by the same supplier. This, obviously, makes identification inspection much more difficult. Software licenses, upgrade tapes, and configuration files are difficult to manage because of this lack of consistency between vendors. If sparing is to be done by other than the COTS supplier, it can be a complex issue. Nonetheless, the railroad can effectively deal with these problems. Remedies include auxiliary identifiers and decals applied at the time of incoming inspection for inventory control, serialization, configuration control, and accounting.

2.2. Software Configuration Baselines

The concept of baselines is central to an effective SCM program; it is however, not a unique SCM concept. The idea of using a known and defined point of reference is commonplace and is central to an effective management process. The essential idea of baselines is that in order to reach a destination, it is necessary to know your starting point. In order to plan for, approve, or implement a configuration change, it is necessary to have a definition of the current configuration that is to be changed. In configuration management, a configuration baseline is a fixed reference configuration established by defining and recording the approved configuration documentation for a CSCI at a specified time. Configuration baselines represent:

- Snapshots which capture the configuration or partial configuration of a CSCI at specific points in time;
- Commitment points representing approval of a CSCI at a particular milestones in its development;
- Control points that serve to focus management attention.

The following are typical configuration baselines that are established during the life-cycle of a software project.

- Functional Baseline - Established by the acceptance or approval of the software or system specification. This baseline typically corresponds to the completion of the software requirement review.

- Allocated Baseline - Established by approval of the software requirements specification. This baseline typically corresponds to the completion of the software specification review.

- Developmental Baseline - Established by the approval of the technical documentation that defines the functional and detailed design (including documentation of interfaces and databases for the computer software). Normally, this baseline corresponds to the time frame spanning the preliminary design review and the critical design review.

- Product Baseline - Established by approval of the product specification following completion of the last formal functional configuration audit (FCA).

2.3. Software Identifiers

Configuration identification consists of selecting the configuration items and recording their functional and physical characteristics as set forth in technical documentation such as specifications, drawings, and associated lists. The following are examples of items managed in the SCM system:

- Management plans;
- Specifications (requirements, design);
- User documentation;
- Test plans, test design, case, and procedure specifications;
- Test data and test generation procedures;
- Support software used in development, even though not part of the delivered software product;
- Data dictionaries and various cross-references;
- Source code (on machine-readable media);
- Executable code (the run-time system);
- Libraries;
- Databases (data that are processed and data that are part of a program);
- Maintenance documentation (e.g., listings, detail design descriptions).

Effective management of the development of a software product requires careful definition of the configuration items. Changes to these items also need to be defined since these changes, along with the project baselines, specify the evolution of the software product. SCM includes all of the entities of the software product as well as their various representations in documentation. The entities are not all subject to the same SCM disciplines at the same time.

Support software is the class of software that may or may not be delivered with the software product, but is necessary for designing, enhancing, or testing the changes made during the life-cycle of the product. Support software may be user-furnished, developed in-house, leased from a vendor, or purchased off-the-shelf. In SCM, the focus is on describing the necessary controls used to manage support software. Developers and maintainers need to ensure that the support software is available for use for as long as the software product is in use. Whenever a production baseline is established, it is very important to archive all environmental and support tools along with the production code.

Software tools used in development, especially compilers and middleware, should be placed under configuration control so their identities and versions are included in the baseline data about each release.

Consideration also needs to be given to the hierarchy of entities managed during the SCM system. There are several different ways of structuring this hierarchy. One way is a three-level hierarchy consisting of configuration items, components, and units. Another

way of structuring the entities is in terms of the interrelationships between the software being developed and the other software entities used during development and testing such as support software, test software, and the operating system. A third method for structuring entities is in terms of the intermediate products generated in the process of building the software product, such as: modifiable entities (e.g., source code, document, test data); compilation entities (e.g., compilers, support software); and configuration items (e.g., different representations created in the process of producing deliverables).

An important aspect of configuration identification is the use of a formal naming convention for each entity. This naming convention, which typically uses a combination of mnemonic labels and version numbers, should be applied to all components of a configuration item. In establishing the naming convention, consideration should be given to system constraints such as name length or composition. Consistency in the application of this identification scheme is critical to accurate tracking of the software engineering process

For each CSCI, the software identifier consists of a name or other identifier and a version identifier, assigned by the developing contractor. The identifiers relate the software to its associated configuration documentation, revision, and release date. The software and version identifiers are embedded within the source code, and are marked on media containing the software. A method is typically employed to display the identifier and version to the user of the software upon command. Firmware identification is labeled on the device or embedded in the firmware, or if the device is too small, on the next higher assembly. Firmware identification includes the top-level document/drawing that defines how these components fit together for the firmware assembly.

Each accountable copy of a software product (e.g., source code tape), with the exception of the EM and listings, is normally assigned a unique copy number both externally and embedded within the software. For software products that require more than one unit of physical storage per copy, a volume number is assigned to each unit of storage both externally and embedded on the software.

2.4. Software Configuration Control Activities

Software Configuration Control (SCC) is perhaps the most visible element of SCM. It is the process used to manage preparation, justification, evaluation, coordination, disposition, and implementation of changes and deviations to effected CSCIs and base lined configuration documentation. The primary objective of configuration control is to establish and maintain a systematic change management process that regulates life-cycle costs, and:

- Allows optimum design and development latitude with the appropriate degree, and depth of configuration change control procedures during the life-cycle of a CSCI.
- Provides efficient processing and implementation of configuration changes that maintain or enhance operational readiness, supportability, interchangeability and interoperability.

- Ensures complete, accurate, and timely changes to configuration baseline maintained under appropriate configuration control authority.
- Eliminates unnecessary change proliferation.

The span of configuration control begins once the first configuration document is approved and baselined. This normally occurs when the configuration baseline is established for the CSCI's. At that point change management procedures are employed to systematically evaluate each proposed engineering change or requested deviation to the baseline, to assess the total change impact (including costs) through coordination with affected functional activities, to disposition the change or deviation and provide timely approval or disapproval, and to assure timely implementation of approved changes. Configuration control is an essential discipline throughout the program life-cycle.

Through the configuration control process, the full impact of proposed engineering changes and deviations is identified and accounted for in their implementation. The configuration control process ranges from less formal processes to a very disciplined and formal process. The configuration control process is employed to make sure all personnel communicate the correct version of CSCI to be used. In addition, the process makes affected parties aware that a change is being developed and enables them to provide pertinent input.

When managing COTS items, performance specifications (performance baseline) are the key point of control. In fact, they are the only legitimate basis for configuration control that the railroad can use. The railroad does not have rights to the design data of a COTS supplier, and cannot directly change it. The railroad may request the supplier to make a change to its product, but does not have the ability to perform that change if the supplier is not in agreement. Selection of a COTS item is based in part on life-cycle cost considerations; the railroad should be cautious about obviating the cost benefit by attempting to over-control the supplier. The railroad also can choose not to use the supplier's product. The supplier has complete configuration control over the COTS product. The supplier may offer changes (improvements, added features) that are optional (perhaps at extra cost) at any time. On the other hand, the supplier may make configuration changes to the product for competitive reasons without any knowledge or compliance by the integrator.

COTS suppliers are also subject to unannounced changes by their own suppliers, which may in turn result in changes to the COTS product design. These supplier initiated changes often improve the product, but are not always made with appropriate modification of technical data or in concert with programmed change activity of the ultimate end user. Considering the nature of the respective end items, the supplier's standard practices, and the competitive environment, requirements for configuration control will vary somewhat from supplier to supplier. Wherever possible, railroad to COTS supplier configuration control requirements should include the following as a minimum:

- Advance notification of software changes that may impact the baseline;
- Advance notification of pending obsolescence;
- Advance notification of changes to versions and releases.

The railroad can be the recipient of short-term notice of changes/obsolescence, and may be forced into a reactive mode. Without direct control of the product evolution, the railroad must compensate by being aware of pending changes as early as possible and performing change impact analyses that assess alternate solutions to determine what action is in their best interests. The impact is minimized by anticipating the likely level of change activity that will occur, including redesign efforts to compensate for unplanned COTS iterations.

2.5. Software Configuration Status Accounting

Software Configuration Status Accounting (SCSA) is the process of creating and organizing the knowledge base necessary for the performance of software configuration management. In addition to facilitating SCM, the purpose of SCSA is to provide a highly reliable source of configuration information to support all software development, maintenance, logistic support, modification, and maintenance. It is constrained only by contractual and business provisions that establish the program life-cycle phase, tasks to be performed, and the railroad organization (or contractor) tasked to perform them. In addition to the use of automated configuration management tools, the process is aided or facilitated by the documented CM processes and open communications between all personnel involved in handling the software. The outputs from this activity provide visibility into the CSCI, its status, and its configuration information at all locations. SCSA also include "metrics" developed from the information collected in the SCSA system and "management prompts" resulting from analysis of the software configuration management database.

2.6. Configuration Verification and Audit

The configuration verification and audit process includes:

- Configuration verification of the initial configuration of a CSCI, and the incorporation of approved engineering changes, to assure that the CSCI meets its required performance and to documented configuration.
- Configuration audit of configuration verification records and physical product, to validate that a CSCI has achieved its performance requirements and matches the configuration documentation for the system/CI being audited, is consistent with the product meeting the requirements.

The common objective is to establish a high level of confidence in the configuration documentation used as the basis for configuration control and support of the product throughout its life-cycle. Configuration verification is a process that is common to configuration management, and quality assurance. It is the means by which the railroad verifies the actual installation of the CSCI's configuration and establishes the as-built configuration. Configuration verification should be an imbedded function of the process for creating and modifying the CSCI at each location. Successful completion of verification and audit activities results in verified CSCIs and a documentation set that may be confidently considered a product baseline. It also results in a validated process to maintain the continuing consistency of product to the baseline. Configuration verification is an on-going process. The more confidence in a contractor's configuration

verification process, the easier the configuration audit process becomes. The reward for effective release, baselining, and configuration/change verification is delivery of a known configuration that is consistent with the documentation and that meets its performance requirements. These are precisely the attributes needed to satisfy the ISO-9000 series requirements for design verification and design validation as well as the ISO-10007 requirement for configuration audit.

The dictionary definition of the word "audit" as a final accounting gives some insight into the value of conducting configuration audits. As has been discussed earlier, configuration management is used to define and control the configuration baselines for the CSCIs and the system. In general, when the CSCIs are baselined, an audit is done to ascertain that the baseline information is an accurate representation of the field installation. The operation and life-cycle support of the CSCI is based on this documentation. To fail to assure its accuracy can complicate future logistics support of the CSCI, as well as result in violations of Federal regulations for non-compliance with the SMCP. Configuration audits provide the framework, and the detailed requirements, for verifying that the field deployment successfully matches the configuration.

There are two general categories of audits - Functional Configuration Audit (FCA) and Physical Configuration Audit (PCA.) FCAs are usually conducted after a major change or a significant number of minor changes have occurred, or before the establishment of the product baseline. The FCA records differences between the SRS and the CSCI under audit. PCAs ensure that the as-built configuration is accurately reflected by the released documentation to establish the product baseline. The released engineering documentation and quality control records are verified to make sure the as-built or as-coded configuration is reflected by this documentation. PCAs are usually conducted concurrently with FCAs, or immediately following an FCA.

2.7. Organization & Functions

The SCM organization will vary depending upon the scope and complexity of the system. SCM interfaces with the functions listed below to control software configuration and release activities. Depending on the size of the organization, the functional groups defined below may be combined (e.g., the Software Systems Engineering Group and the Software Design and Development Group may be one group known as software development.

a. Program Management - Responsible for and has the authority to ensure complete fulfillment of all program requirements. The Program Manager has the overall responsibility for acquisition, funding, and transitioning of the project.

b. Project Management - Responsible for the technical aspects of the project. The Project Manager has the responsibility for local funding, allocations, scheduling, tasking, and reporting to program management.

c. Software Systems Engineering - Responsible for systems design (and associated documentation), overview, and guidance; detailed design and coding; test plans, procedures, and reports; software unit testing; and, preliminary CSCI testing.

d. Software Design and Development - Responsible for software design (and associated documentation), overview, and guidance; detailed design and coding; test plans, procedures, and reports; software unit testing; and, preliminary CSCI testing.

e. Software Test - Responsible for the conduct of software testing, including preparation of test plan, description, procedures, and reports. The Software Test Group ensures that the correct configuration is undergoing test and incorporates approved changes into released test documentation based on change request baselining data from SCM. The Software Test Group confirms verification of change request corrective measures prior to change request closure. SCM identifies all change requests included in an Engineering Master (EM) that is to be tested. Test personnel then provide SCM a copy of the test report.

f. Software Quality Assurance (SQA) - Responsible for auditing the software development activities and products (FCA and PCA) and certifying of SCM compliance with this plan and DTPs.

g. System Test - Responsible for administering the verification and validation (V&V) testing prior to release of the software. The System Test Group is a separate organization from the Software Development Group (i.e., the Software Systems Engineering Group and the Software Design and Development Group).

h. Logistics - Responsible for ensuring that changes made to a system are supportable. SCM provides CSCI and associated technical data for logistics evaluation.

i. Data Management (DM) - Responsible for the receipt, distribution, and tracking of technical data associated with the project.

SCM is responsible for maintaining configuration control over software developmental configurations and baselines and for processing changes to the software configuration. SCM functions include Software Development Library (SDL) operation, software product release coordination, and change request processing and tracking.

The responsibilities of each SCM function are listed in the paragraphs below.

2.7.1. Configuration Identification.

a. Establish methods and procedures for unique identification of CSCIs.

b. Establish and maintain functional, allocated, and product baselines and the developmental configuration (identify, document, archive, and track changes to system releases).

c. Establish and follow release procedures to obtain product baselines for new version releases.

d. Coordinate assignment of identifying numbers for CSCIs and documents.

e. Provide documentation that reflects the release software package.

f. Coordinate release of software and associated documentation to release organizations.

g. Maintain records and prepare reports on release coordination activities.

2.7.2. Configuration Control.

a. Serve as a member of the Software Configuration Control Board (SCCB). SCM is responsible for preparing and distributing the meeting agenda and minutes and for recording action items and their resolution.

b. Establish and document configuration change control procedures.

c. Establish and follow configuration controls for software and documentation.

d. Place contents of baseline and developmental configurations under configuration control in the SDL.

e. Generate executable load modules from controlled source code.

f. Ensure that the contents of the SDL are changed by SCM personnel and only upon receipt of the appropriate paper work signed by the SCM manager.

g. Prepare and maintain master(s) of the currently active version of each CI until superseded by a new version. Retain superseded versions of the master(s) in the SDL archive files.

h. Maintain records and prepare reports on SDL activities and software products.

i. Perform non-technical check of software documentation.

j. Interface with Software Change Review Board (SCRB) chairperson to schedule SCRB meetings, prepare SCRB agendas, and record SCRB meeting minutes.

2.7.3. Configuration Status Accounting (CSA).

a. Provide CSA recording and reporting.

b. Maintain accounting of software changes by tracking change requests, ensuring traceability to a formal change proposal (i.e., ECP) from initiation through resolution and disposition.

c. Prepare status reports on change requests, formal change proposals (i.e., ECPs), and changes.

2.7.4. Configuration Audits.

a. Support requests for audit and certification of software systems by SQA or the independent auditor.

b. Perform reviews of SCM activities and products.

c. Review and update SCM documentation as required to ensure that current applicability is maintained.

2.8. Configuration Control Boards

Configuration Control Boards (CCBs) enable the implementation of change controls at optimum levels of authority and scope. CCBs can exist in a hierarchical fashion (for example, at the program, system design, and program product level), or one board may have authority over all levels of the change process. In most organizations, the CCB is composed of senior level managers. They include representatives from the major software, hardware, test, engineering, and support organizations as defined in the SCM Plan. The purpose of the CCB is to control major issues such as schedule, function, and the configuration of the system as a whole.

The more technical issues that do not relate to issues of performance, cost, and schedule are often assigned to a Software Configuration Control Board (SCCB). The SCCB discusses issues related to specific schedules for partial functions, interim delivery dates, common data structures, design changes, and the like. This is the place for decision-making concerning the items that must be coordinated across configuration items, but they do not require the attention of high-level management. The SCCB members should be technically well versed in the details of their area, while the CCB members are more concerned with broad management issues facing the project as a whole and with customer issues.

Depending on the size and nature of the software project, the SCCB may consist of a single librarian, multiple librarians, or include many SCCBs, each with a different functional responsibility for ensuring that changes are implemented and tested according to standard procedures and that hardware/software interfaces and interfaces between software modules are not violated. The SCCB also focuses on overall project management responsibility for ensuring that design or requirements specifications are not violated and that software changes are implemented according to cost and schedule constraints. The size and structure of SCCBs normally change over the life-cycle of the software.

The Software Change Review Board (SCRB) considers the recommendations of the project's SCCB for final approval or disapproval of proposed engineering changes to a CSCI's current approved configuration and its documentation. The board also approves or disapproves proposed waivers and deviations. The SCRB is responsible for evaluating and approving or disapproving proposed software changes. The evaluation of proposed changes must consider as a minimum such factors as documentation, equipment interfaces, training equipment, implementation costs, and performance requirements.

Proposed changes submitted for SCRB action must be complete with respect to technical requirements, justification, cost information, logistic requirements, interface requirements, retrofit requirements, and other applicable information. When a proposed change affects any system or item under the cognizance of another SCRB, joint SCRB meetings are held as required.

The SCCB supports the Project Manager and is composed of technical and administrative representatives who recommend approval or disapproval of proposed engineering changes to a CSCI's current approved configuration and its documentation. The board also recommends approval or disapproval of proposed waivers and deviations from a CSCI's current approved configuration and its documentation. Issues that the project's SCCB is unable to resolve or that involve a change in scheduling or fiscal costs are initially addressed by the SCCB and forwarded to the program's SCRB for final approval or disapproval and recommendations.

The SCCB has authority for managing the project's software through the performance of the functions listed below.

a. Authorize establishment of software baselines and identification of CSCIs.

b. Represent interests of project management and all groups who may be affected by software changes to the baselines.

c. Assign, review, and provide for disposition of action items.

d. Provide required staff coordination on all proposed or reviewed changes or modifications.

e. Serve as a source for the coordination of software technical expertise for the project.

f. Determine or review the availability of resources required to complete the proposed change or modification, assess the impact of the proposed change upon the system, examine cost considerations, and determine the impact of the change on development and test schedules.

g. Monitor the design, production, and validation process for approved modifications, and initiate, when required, the corrective actions necessary to ensure design compatibility and integrity, cost-effectiveness, and conformance to scheduled milestones.

h. Direct software change implementation on changes approved by the SCCB.

i. Exercise interface management support and control for project software.

2.9. Baseline Change Process

Changes occur at all phases in the software life-cycle. Design or implementation changes may be necessary if requirements change, or when deficiencies are identified in the design or implementation approach to a stable requirement. Testing may uncover defects that require changes in the code or the design and requirements. Changes must be made to the right version of the code, testing must verify performance of the change and the integrity of the remaining software, and all associated documentation must be updated to be consistent with the final code.

A mechanism is needed to process change requests (e.g., discrepancies, failure reports, requests for new features) from a variety of sources throughout the development process and during operation and support of the software product. This mechanism should extend to a definition of a process to track, evaluate, and implement change requests into a new version and new release of the software. Generally, no single procedure can meet the needs of all levels of change management and approval levels.

The minimum activities needed for processing changes include:

- Defining the information needed for approving the requested change.
- Identifying the review process and routing of information.
- Describing the control of the libraries used to process the change.
- Developing a procedure for implementing each change in the code, the documentation, and the released software product.

The baseline change process is a continuous function that involves the preparation, implementation, and distribution of CSCI and associated documentation changes. It involves activity at both the project and program levels. Changes to a baseline configuration are initiated through a change request process that involves the preparation of a defined series of documents (change forms) whose status is determined by a hierarchy of control boards. Change requests are used to report problems and propose changes or enhancements to software or documentation. A change request must be documented, submitted, reviewed, and approved prior to implementation. Change requests against developmental baselines are resolved by the SCCB. Change requests against established baselines require approval of the SCRB.

Configuration changes to established baselines are categorized as either Class I or Class II. A Class I change would be required if any of the following were affected:

a. The functional, allocated, or product configuration documentation.

b. Complete schedules.

c. Any of the following contractual factors:

   (1) Cost to including incentives and fees.

   (2) Contract guarantees or warranties.

(3) Contractual deliveries.

(4) Scheduled contract milestones.

All Class I changes should be processed via an ECP. All changes to CSCIs that do not meet one or more of the Class I change requirements are identified as Class II changes. Examples of Class II changes include the following:

a. Changes to correct editorial errors.

b. Additions to clarifying notes or diagrams.

c. Changes to hardware that do not affect any Class I factors.

Generally, the following change forms are used for control of software baselines:

a. Engineering Change Proposals (ECPs)

b. Specification Change Notices (SCNs)

c. Notices of Revisions (NORs)

d. Deviation and Waiver

The ECP is used to document all proposed changes to established baselines. The completed ECP must include detailed descriptions, justifications, and costs for the proposed change. The SCN is used to correct or update specifications. Proposed SCNs are submitted with Class I ECPs and provide proposed text changes to applicable specifications. The SCN identifies the document to be changed, the SCN number, its status (proposed or approved), the related ECP, and other identifying data. The NOR is primarily intended for use when the master drawing list and other documents comprising the configuration identification are not held by the originator of the ECP. NORs permit the ECP previewing or approving activity to direct the custodian of an applicable document to make specific revisions in affected documents.

Changes for a CCB are always prioritized. Table 1 provides commonly used prioritization scheme

Table 1: EXPLANATION OF PRIORITIES.

PRIORITY	APPLIES IF A PROBLEM COULD:
1	a. Prevent the accomplishment of an operational or mission essential capability. b. Jeopardizes safety, security, or other requirement designated "critical".
2	a. Adversely affect the accomplishment of an operational or mission essential capability and no work-around solution is known. b. Adversely affect technical, cost, or schedule risks to the project or to life-cycle support of the system, and no work-around solution is known.
3	a. Adversely affect the accomplishment of an operational or mission essential capability but a work-around solution is known. b. Adversely affect technical, cost, or schedule risks to the project or to the life-cycle support of the system, but a work-around solution is known.
4	a. Result in user/operator inconvenience or annoyance but does not affect a required operational or mission essential capability. b. Result in inconvenience or annoyance for development or support personnel, but does not prevent the accomplishment of those responsibilities.
5	Result in any other affect.

2.10. Reports and Records

The techniques and methods used for implementing control and status reporting in SCM generally center around the operation of software libraries. Software libraries are a controlled collection of software and related documentation designed to aid in software development, use, and maintenance. Software libraries provide the means for identifying and labeling baselined entities, and for capturing and tracking the status of changes to those entities.

Libraries have been historically composed of documentation on hard copy and software on machine-readable media, but the trend is moving toward all information being created and maintained on machine-readable media. This trend, which encourages the increased use of automated tools, leads to higher productivity. The trend also means that the libraries are a part of the software engineering working environment. The SCM functions associated with the libraries have to become part of the software engineering environment, making the process of configuration management more transparent to the software developers and maintainers.

The number and kind of libraries will vary from project to project or organization to organization, according to variations in the access rights and needs of their users, which are directly related to levels of control. The items maintained in the libraries may vary in physical form based on the level of technology of the software tools. When the

management of the libraries is automated, the libraries that represent different levels of control may be functionally (logically) different even though they are physically the same. The insertion of entities and changes to entities in a controlled library should produce an auditable authorization trail.

The names of libraries may vary, but fundamentally there are three kinds: dynamic, controlled, and static. The dynamic library, sometimes called the programmer's library, is a library used for holding newly created or modified software entities (units/modules or data files and associated documentation). This is the library used by programmers in developing code. It is freely accessible to the programmer responsible for that unit at any time. It is the programmers' workspace and is usually controlled by the programmers.

The controlled library, sometimes called the master library, is a library used for managing the current baseline(s) and for controlling changes made to them. This is the library where the components and units of a configuration item that have been promoted for integration are maintained. Programmers and others can freely make copies for use. Changes to components or units in this library must be authorized by the responsible authority (which could be a configuration control board or other body with delegated authority).

The static library, sometimes called the software repository, is a library used to archive various baselines released for general use. This is the library where the master copies plus authorized copies of software configuration items that have been released for operational use, are maintained. Copies of these masters may be made available to requesting organizations.

SCM has the prime responsibility for managing, compiling, maintaining, and publishing the detailed software CSA reports. These reports provide the status to management that all changes between the software technical description and the software itself are being accounted for on a one-to-one relationship. This status information, together with the CSA reports maintained by the SCM organization, is an input for the final review for product acceptance. Project management determines the frequency of distribution and recipients of the CSA reports. These reports include the information listed below.

a. Identification of currently approved configuration documentation and configuration identifiers associated with each CSCI.

b. Status of proposed change requests from initiation to implementation.

c. Results of configuration audits, and status and disposition of discrepancies.

d. Traceability of changes from baselined documentation of each CSCI.

e. Effectiveness and installation status of configuration changes to all CSCIs at all locations.

The above reports answer basic questions regarding the approved configuration (baseline) and the implementation status of changes to the baseline. Requests for CSA reports originating outside the project are directed for approval to Project Management, which authorizes need-to-know access.

The records maintained by SCM contain detailed data that documents that the as-built software conforms to its technical description and specified configuration. They include the information listed below.

a. Approved technical documentation for each CSCI.

b. Status of proposed changes.

c. Implementation status of approved changes.

d. Status of software problems.

e. A record of change request status.

2.10.1. Change Request Records

The change request records contain a record of all change requests and related information. It includes, but is not limited to, the data listed below.

a. Change request number.

b. Title.

c. Date.

d. Software product name or acronym.

e. Part number or revision in error.

f. Originator.

g. Change source (e.g., ECP), if applicable.

h. Current change request status.

i. Change request disposition.

2.10.2. Library(ies) Inventory Records

The library inventory records contains a record of each software product stored in the library(ies). It includes, but is not limited to, the data listed below.

a. Product name.

b. Part or document number and revision.

c. Date of creation, last modification, and last access.

d. "Master" or "Copy" designation.

e. Authorizing paperwork type and number.

f. Type of media.

g. Location.

h. Classification.

2.10.3. Data Distribution Records

The data distribution records contain a record of all data (e.g., documents and drawings) distributed by the software organization through DM. The table includes, but is not limited to, the information listed below.

a. Type and identification number of distribution request.

b. Date of submittal.

c. Media identification.

d. Reason for distribution.

e. Classification.

2.10.4. Release Records

The release records contain a record of all releases made by the software organization (e.g., drawings, documents, software documents, tape). It includes, but is not limited to, the information listed below.

a. Date of release.

b. Type of release.

c. Software product released.

d. Changes incorporated into the release.

e. Approval signatures.

f. Location of masters.

2.10.5. Archive Records

SCM maintains a record of all archived material. Archived material includes obsolete material and data not required for current use and off-site stored backup data in case of loss of on-line data.

2.11. Automated Tools for Software Configuration Management

The SCM automated tools used for a project and described in the SCM Plan need to be compatible with the software engineering environment in which the development or maintenance is to occur. SCM tools offer a wide range of capabilities and choices have to be made as to the tool set most useful for supporting the engineering and management environment.

Configuration management of complex software may depend on ad hoc systems whose effectiveness is based on close adherence to written procedures. Such systems, which may be paper-based or electronic, require considerable effort to achieve the necessary levels of integrity and traceability, particularly for team development projects. Software tools designed for configuration management can simplify this process and provide better access to the information required by the developer, project manager, or user.

Software tools for configuration management can provide a number of benefits over paper-based systems, including:

- Increased reliability and repeatability of the development process through automation of many mundane processes.
- Ability to track and manage changes within a user-defined process workflow.
- Enhanced productivity, by eliminating wasted effort and speeding up frequent activities such as the build and support of concurrent development.
- A means to manage product variants.
- Real-time analysis of all development activities.
- Complete audit trails of both software and documentation.
- More effective team management.

The automated tools described in this section are classified into broad categories in terms of the level of automation they provide to the project.

2.11.1. Basic Tool Set

The basic tool set is compatible with an environment that is relatively unsophisticated. The tools control the information on hard copy regarding a program product. These tools provide a capability that distinguishes between controlled and uncontrolled units or components. The tools simplify and minimize the complexity, time, and methods needed to generate a given baseline.

The basic tool set includes:

- Basic database management systems.
- Report generators.
- Means for maintaining separate dynamic and controlled libraries.
- File system for managing the check in and check out of units, for controlling compilations, and capturing the resulting products.

2.11.2. Advanced Tool Set

The advanced tool set provides a capability for an SCM group to perform more efficiently on larger, more complex software engineering projects. These tools provide an environment that has more computing resources available. It provides the means of efficiently managing information about the units or components and associated data items. It also has rudimentary capabilities for managing the configurations of the product (building run-time programs from source code) and providing for more effective control of the libraries.

The advanced tool set includes:

- Items in the basic tool set.
- Source code control programs that will maintain version and revision history.
- Tools for comparing programs for identifying (and helping verify) changes.
- Tools for building or generating executable code.
- A documentation system (word processing) to enter and maintain the specifications and associated user documentation files.
- A system/software change request/authorization (SCR/SCA) tracking system that makes requests for changes machine-readable.
- Capability to manage concurrent development efforts.

2.11.3. Online Tool Set

The online tool set requires an interactive programming environment that is available to the project. It provides an organization with the minimal SCM capabilities needed to support the typical interactive environment currently available in industry. These tools also provide online access to the programming database and the resources necessary for using the tools.

The online tool set includes:

- Generic tools of the advanced tool set integrated so they work from a common database.
- An SCR/SCA tracking and control system that brings generation, review, and approval of changes online.
- Report generators working online with the common database, and an SCR/SCA tracking system that enables the SCM group to generate responses to online queries of a general nature

2.11.4. Integrated Tool Set

The integrated tool set integrates the SCM functions with the software engineering environment so that the SCM functions are transparent to the engineer. The software engineer becomes aware of the SCM functions only when he/she attempts to perform a function or operation that has not been authorized (for example, changing a controlled entity when the engineer does not have the required level of authority or control).

An integrated tool set includes:

- Online SCM tools covering all functions.

- An integrated engineering database with SCM commands built into the online engineering commands commonly used in designing and developing programs (most functions of CM are heavily used during design and development phases).
- The integration of the SCM commands with online management commands for building and promoting units and components.

2.11.5. Tool Selection

SCM tools should be carefully selected to match working practices and ideology. Selection of SCM tools should be based on the following features.

- Cross-platform support.
- Developer empowerment (librarian or other supervisory role optional).
- Match to existing work practices (task or life-cycle based).
- Tool integration (between components and other software engineering tools).
- Ease of installation and use.
- Code visibility outside tool (to support intranet technology).
- Supplier support before and after delivery.
- Market position of supplier and product.
- Overall cost.

3. Typical Software Management Control Plan Contents

The Software Management Control Plan (SMCP) defines all of the procedures, organizational responsibilities, and tools to be used within the SCM process. As such, it must either include well-developed, detailed procedures or refer to their locations in other documents. A SMCP is not a one-size-fits-all document. Rather, it must be tailored to meet the needs of the organization. It also is important to note that the size of the SMCP should not be so large as to intimidate users. If the procedures are sizeable, they should be broken out into their own documents to reduce cultural resistance.

Performing SCM for the sake of SCM becomes a cumbersome, time-consuming process, which does not result in benefits. A concerted effort to tailor the process must be made. If this is done, the following benefits can be obtained.

- The plan documents the SCM process and as such acts as the tool used to gain project and management support for the process.
- The plan forces an organization to define and describe the process.
- The plan causes the organization to think about what it will do and how it will do it.
- The plan serves as a contract vehicle (in some cases).

Planning for SCM is essential to its success. The routine clerical-type functions associated with SCM are repetitious and can be automated fairly easily. The more important disciplines of SCM, such as defining a scheme for identifying the configuration items, components, and units; or the systematic review of changes before authorizing their inclusion in a program, are activities that require engineering judgment. Relating engineering judgment with management decisions, while also providing the necessary

clerical support without slowing the decision-making process, is the critical role of SCM planning. Effective SCM involves planning how all of these activities are to be performed, and performing the activities in accordance with the plan.

The planning and application of SCM is very sensitive to the context of the project and the organization being served. If SCM is applied as a corporate policy, it must be done in such a way that the details of a particular SCM system are reexamined for each project (or phase for very large projects). It must take into consideration the size, complexity, and criticality of the software project being managed; and the number of individuals, amount of personnel turnover, and organizational form and structure.

3.1. Developing A Plan

The Software Engineering Institute at Carnegie Mellon University identified 10 elements to successfully implementing software configuration management. They are: planning, process, culture, people, product, automation, management, the SCM plan, the SCM system, and the SCM adoption strategy. These elements are:

1. Determining what issues need to be documented in the plan and resolving them (Planning)
2. Specifying how to implement the plan (Process)
3. Picking a solution that matches the way the organization does business (Culture)
4. Identifying who is involved in implementing and executing the plan (People)
5. Determining what comes under SCM (Product)
6. Determining what tools will be used to help (Automation)
7. Getting buy-in and backing (Management)
8. Documenting decisions (SCM Plan)
9. Implementing (SCM System)
10. Bringing the system online and populating the data (SCM Adoption Strategy)

Of all of these, Carnegie Mellon found that actually writing the plan (Documenting decisions) was the least difficult step in the whole process. It was the other 9 steps that caused all of the problems. The difficult part was determining the answers for steps 1 through 6, obtaining management buy-in (step 7), and then executing steps 9 and 10.

First, recognize that the development of an effective SMCP is not a turnkey job that can be completely handed over to a consultant. It is essential to have organizational staff actively involved with, or leading the development of, the plan because organizational staff has the best understanding of the system functionality and change control needs. Developing a SMCP is not a mysterious, magical process. Rather, it simply requires concerted effort and communication to ensure that the plan meets the needs of the organization in question.

The best way is to start is with a standard, such as IEEE Std 828-1990, IEEE Standard for Software Configuration Management Plans, American National Standards Institute, 1990. or IEEE Std 1042-1987, IEEE Guide to Software Configuration Management, American

National Standards Institute, 1987 as a template, and then customize that to the organizations specific needs. That will then guide development of the various procedures and processes. Doing the customization is by far the most difficult step in the process. The organization must define and document those procedures that will work for their organization.

The Plan is a Living Document – It Will Change Too. The plan will need to change to reflect the changing environment. As an organization gains experience, this experience will dictate changes to the plan. Experience will determine what works and what does not work for a given situation. Clearly, the SMCP subsequently will change throughout the system's life-cycle. For this reason, the SMCP is subject to change control and should be treated as any other component of the SMCS.

In general, a sufficient plan must address at least the following six topics:

1. Introduction - Description of the plan's purpose, scope of application, key terms, and references.

2. SCM Management (Who?) - Identifies the responsibilities and authorities for accomplishing the planned activities.

3. SCM Activities (What?) - Identifies all activities to be performed as applying to the project.

4. SCM Schedules (When?) - Identifies the required coordination of SCM activities with the other activities in the project.

5. SCM Resources (How?) - Identifies tools and physical and human resources required for execution of the plan.

6. SCM Plan Maintenance - Identifies how the plan will be kept current while in effect.

3.2. Tailoring a Plan

SCM disciplines should be practiced as a part of every software engineering project. The completeness and level of detail for the configuration management disciplines depend on the size, complexity, and importance of the project.

The level of formality associated with SCM depends on the risks to be mitigated. Using a broad definition of SCM as those processes capturing a logical snapshot of a dynamic development process, the formality can range from very informal to fully controlled with auditable processes. Software with minimal risks associated with loss of historical or baseline information may have no formally defined process. For instance, the software developer may decide which versions should be kept based on a personal risk assessment. At the other end of the formality scale, large projects introduce processes to assure the capture of a useful set of final and intermediate deliverables with rules for changes to the

deliverables. As there is a cost to the implementation of formal procedures, it is important to select processes that minimize risks proportioned to their costs.

The disciplines of SCM apply to the development of programmed logic, regardless of the form of packaging used for the application. Whether software is released for general use as programs (e.g., RAM or DRAM) or embedded in read-only memory (ROM), it is a form of logic. Therefore, SCM disciplines can and should be extended to include development of the software's component parts (e.g., source code and executable code).

Firmware raises some special considerations for configuration management. While being developed, the disciplines of SCM apply, but when made part of the hardware (e.g., burned into [EP]ROM or [EEP]ROM), the disciplines of hardware configuration management apply. Testing may vary, but the SCM requirements are generally the same. The packaging of [EP]ROM or [EEP]ROM versus RAM or DRAM code also introduces and necessitates different identification procedures**.**

3.3. Sample Plan Outline

As an example, a fully developed SMCP outline follows:

1 Introduction
1.1 Purpose
1.2 Scope
1.3 Definitions and Abbreviations
1.3.1 Definitions
1.3.2 Abbreviations
1.4 Plan Implementation
1.4.1 Implementation Schedule
1.4.2 Required Resources
1.4.3 Implementation Risks
1.4.4 Cost Estimates
1.4.5 Change Procedures (and History)
1.5 References

2 Management Organization & Documentation
2.1 Organizational Unit and Interfaces
2.1.1 Operations Department
2.1.2 Signal/Communications Department
2.1.3 Mechanical Department
2.1.4 Information Technology Department
2.2 Activities and Organization Unit Responsibilities
2.2.1 Configuration Identification
2.2.2 Configuration Control
2.2.3 Configuration Status Accounting
2.2.4 Configuration Audits and Review
2.2.5 Configuration Control Board

2.2.6 Interface Control
2.2.7 Quality Assurance
2.3 Software Configuration Management Documentation
2.3.1 Configuration Identification Documentation
2.3.2 Configuration Control Board Documentation
2.3.3 Configuration Status Accounting Documentation
2.3.4 Schedules and Reports for Reviews and Audits
2.3.5 Configuration Baseline Documentation
2.3.6 Interface Control Documentation
2.3.7 Quality Assurance Documentation
2.4 Configuration Management, Development, and Maintenance Tools
2.5 Other Applicable management Policies, Directives, and Procedures
2.5.1 Polices
2.5.2 Directives
2.5.3 Procedures

3 Software Configuration Management Activities
3.1 Configuration Identification Processes and Procedures
3.1.1 Required Documentation
3.1.2 Identification of Computer Software Configuration Items
3.1.3 Configuration Identification of Deployed System Baselines
3.1.4 Inspection and Receiving
3.2 Configuration Control
3.2.1 Reporting of Changes
3.2.2 Change Proposal Processing
3.2.3 Level of Authorities for Approvals
3.2.4 Configuration Control Board Procedures
3.3 Auditing
3.3.1 Verification Processes and Procedures
3.3.2 Validation Processes and Procedures
3.4 Interface Control Processes and Procedures
3.5 Quality Assurance Processes
3.5.1 CSCI QA
3.5.2 SCMP QA
3.6 Development and Maintenance Tools
3.6.1 Development
3.6.2 Operations and Maintenance
3.6.3 Configuration management

4 Training Requirements
4.1 Field Personnel
4.2 Office Personnel
4.3 Vendor/Subcontractor/Supplier
4.4 Configuration Management of Training

5 Supplier/Vendor/Subcontractor
5.1 Railroad Subcontract/Vendor Management Organization and Interface
5.2 Required Reports
5.3 Activities and Responsibilities
5.3.1 Compliance with Railroad SCM Requirements
5.3.2 Railroad Intellectual Properties Agreements
5.3.3 Vendor Version Control and Reporting
5.3.4 Patch/Release/Version Upgrade Procedure
5.3.5 CCB Relationships
5.3.6 QA

6 Record Collection and Retention
6.1 Required Records
6.2 Retention Requirements
6.2.1 Manual
6.2.2 Electronic

3.4. Review Criteria

A well-written SMCP should be able to provide affirmative answers/explanation to each of the following questions.

- Does the plan create procedures to ensure identification and control of all configuration items and baselines, including necessary changes to those items?

- Does the plan identify appropriate tools or methods to support the configuration system, including change control and methods of backup?

- Does the plan document procedures for review and authorized release of products consistent with the level of testing applied?

- Does the plan develop a mechanism for coordinating the updating of software at all customer locations?

- Does the plan create procedures for replication and subsequent verification and product identification activities?

- When a configuration management software tool is employed, does the plan clearly document the use of that tool?

- Does the plan develop a mechanism to support the labeling or other means of identification of third-party supplied products including, where necessary, integration into the configuration management system?

- Does the plan develop a mechanism to ensure that software, or designs produced or modified externally to the organization (for example by a subcontractor), are fully integrated into the configuration control process?

- Does the plan determine how the software configuration management system will be tailored to accommodate the size and complexity of the project?

If, after review of a SMCP document, the preceding questions cannot be affirmatively answered/explained, then the plan does not adequately address the minimum requirements of Title 49 CFR § 236.18.

3.5. Recurring Maintenance Review

Maintenance of the SMCP throughout the life-cycle of the various software products is especially important as the disciplines of identification; configuration control, status reporting, and release processing apply throughout the maintenance part of the life-cycle. Differences may be expected in how change processing is managed, and these need to be understood by all participants.

A review of the SMCP should be periodically performed to assess the effectiveness of the approach and the extent to which configuration management procedures are being followed by project staff. This enables adjustments to the SMCP to improve the staff's ability to follow the procedures and allows for more effective approaches to be incorporated as they are developed.