For additional information please contact the
Software Program Managers Network
(703) 521-5231 • Fax (703) 521-2603
E-Mail: spmn@aol.com
http://www.spmn.com

Little Book of Configuration Management

November 1998

THE AIRLIE SOFTWARE COUNCIL

This publication was prepared for the

Software Program Managers Network
4600 North Fairfax Drive, Suite 302
Arlington, VA 22203

[illegible]

Norm Brown
Director, Software Program Managers Network

This work was created by Computers & Concepts Associates, a division of Integrated Computer Engineering, Inc., in the performance of Federal Systems Integration and Management Center (FEDSIM) Contract Number GSOOT98AJD0047 for the operation of the Software Program Managers Network (SPMN).

This guidebook is one in a series of guidebooks published by the Software Program Managers Network (SPMN). Our purpose is to identify best management and technical practices for software development and maintenance from the commercial software sector and to convey these practices to busy program managers and practitioners. Our goal is to improve the bottom-line drivers of software development and maintenance—cost, productivity, schedule, quality, predictability, and user satisfaction.

The Airlie Software Council was convened by a Department of the Navy contractor in 1994 as a focus group of software industry gurus supporting the SPMN and its challenge of improving software across the many large-scale, software-intensive systems within the Army, Navy, Marine Corps, and Air Force. Council members have identified principal best practices that are essential to managing large-scale software development and maintenance projects. The Council, which meets quarterly in Airlie, Virginia, is comprised of some 20 of the nation’s leading software experts. These little guidebooks are written, reviewed, generally approved and, if needed, updated by Council members through the intermediary of the contractor. Your suggestions regarding this guidebook, or others that you think should exist, would be much appreciated.

THE PURPOSE OF THIS LITTLE BOOK

A project's success is highly dependent on effective Configuration Management (CM). CM becomes essential as the size of the software increases. CM is necessary to enable a large team to work together in a stable environment, yet still have the flexibility that's needed to do creative work. All too often, CM is viewed only as a costly and time-consuming effort to be either ignored or thrown together at the last minute. However, not doing CM guarantees a project will be plagued by chaos, errors, permanent damage, low productivity, and unmanageable software evolution.

CM can be defined as stabilizing the evolution of software products at key points and then controlling change. The project that implements good CM can reap enormous benefits. CM can actually enhance productivity by substantially reducing the largest consumer of labor—rework. CM enables visibility into the status of fixing problems and implementing changes. That's all essential information for better forecasting those ever slippery release dates.

There are numerous industry and military standards (i.e., MIL-STD-973 or MIL-STD-2549) that can be referenced as guides for establishing an effective and efficient CM process. These standards should be examined and tailored to the project's specific requirements. Instituting CM best practices is essential to the successful development and maintenance of a software product. The principles and guidelines described in this guidebook will assist you in developing a solid Configuration Management program.

Norm Brown

Norm Brown
Executive Director

INTRODUCTION

Configuration Management is the means by which the content, change, or status of shared information within a project is managed and controlled. A project's success is highly dependent on good CM, and the way CM is performed can make or break a project. By following the straightforward guidelines and disciplines in this book, establishing good CM processes and procedures early in your project, and getting buy-in from all affected parties, you will put your project firmly on the road to success.

[illegible]

Michael W. Evans
President, Integrated Computer Engineering, Inc.

[illegible]

Jeannine Gathmann-Hobbs
Manager, Configuration Management
Lockheed Martin Corporation

THE CM PROCESS

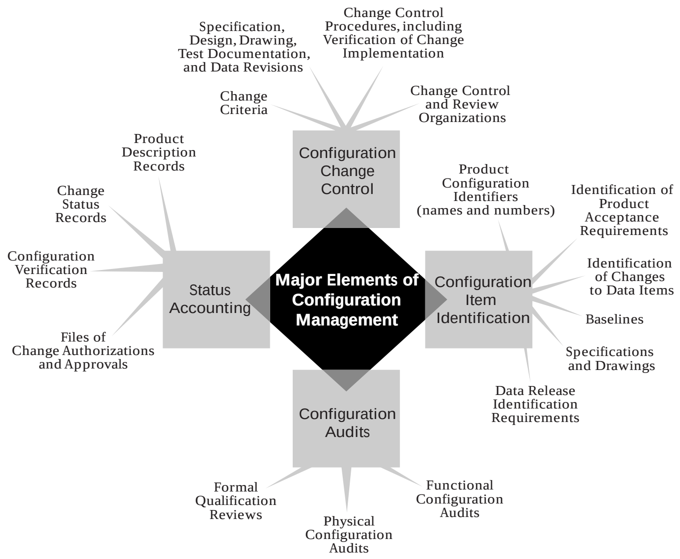
The fundamental purpose of Configuration Management (CM) is to establish and maintain the integrity and control of software products throughout a project's life cycle. This includes products such as performance requirements, functional and physical attributes, and design and operation information. CM is a discipline applying both technical and administrative direction for the control of change and integrity of the product data and documentation. CM involves identifying the configuration of the software (i.e., software work products) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the project's life cycle.

The Software Configuration Management process is comprised of the following integrated activities:

- Configuration identification of artifacts/work products used or developed by a project
- Configuration change control of information, including the impact of changes to organizations, management practices, schedules, budgets, technical or assurance activities, testing or retest requirements, or project status
- Status accounting of artifacts/work products used in the development, release, and maintenance of a project
- Configuration reviews and audits that assess the status and acceptability of products controlled or released by CM

THE CM PROCESS (cont.)

- Project delivery and release management procedures, and the capability to monitor the status of project information
- Establishment of a Software Development Library (SDL) and maintaining the integrity of the work products placed under CM control to ensure repeatability of the products and baselines



Major Elements of Configuration Management
Configuration Change Control
Specification, Design, Drawing, Test Documentation, and Data Revisions
Change Control Procedures, including Verification of Change Implementation
Change Criteria
Change Control and Review Organizations
Configuration Item Identification
Product Configuration Identifiers (names and numbers)
Identification of Product Acceptance Requirements
Identification of Changes to Data Items
Baselines
Specifications and Drawings
Data Release Identification Requirements
Configuration Audits
Formal Qualification Reviews
Physical Configuration Audits
Functional Configuration Audits
Status Accounting
Product Description Records
Change Status Records
Configuration Verification Records
Files of Change Authorizations and Approvals

Figure 1. Major Elements of Configuration Management

CM BASICS

Configuration Management is the basic project control mechanism that establishes and maintains the integrity of software products through the project's life cycle. CM provides:

- ***Configuration Identification***—The ability to identify what information has been approved for concurrent use in the project, who owns the information, how the information was approved for CM control, and the latest approved release.
- ***Configuration Control***—The configuration control process and procedures designating the level of control through which each work product must pass (for example, author control, project-level control, acquirer control); identifying the persons or groups with authority to authorize changes and to make changes at each level (for example, the programmer/analyst, the software lead, the project manager, the acquirer); and the steps to be followed to obtain required authorization for changes, to process change requests, to track changes, to distribute changes, and to maintain past versions. Change control provides the mechanism to build software systems for tests that have a known configuration and can be exactly reproduced.
- ***Status Accounting***—Formalized recording and reporting of the established configuration documents, the status of proposed changes, and the status of the implementation of approved changes. Status record information provides an accessible and

current record of the status of each controlled piece of information that is planned to be used, the content of each release from CM, and who has checked out or is working on a piece of information that the test organization plans on accessing through CM.

- ***Reviews and Audits***—Frequent evaluation of the content, baseline integrity, and release integrity of all controlled products to ensure they conform to their configuration documents.

Information to be controlled includes software and its associated documentation; interface requirements and documentation; engineering artifacts resulting from the methods and tools used by the project; trade studies and user requirements, needs, and expectations; management plans; information and reports; project tools and users' manuals; project records and history; test plans, procedures, cases, scenarios, and data; and test tools (anything concurrently used across project organizations or approved for sharing).

WHY CM IS ESSENTIAL

CM is essential to program and software project success for the following reasons:

1. CM is the means by which shared information (whether it is produced, used by, or released from a software development or support activity) is controlled and maintained.
2. CM methods provide a means to identify, track, and control system development from the inception of the concept for the system until it is replaced or retired.
3. Management of baselines and engineering products through CM provides a sustained control of the information as the engineering functions work their way through the process.
4. CM provides visibility into project control change indicators, including CM churn per month, requirements change per month, and number of defects that are open and closed.

Application of CM to a project environment provides the discipline needed to ensure that the system operational concept is consistent with the capability that results.

CM also provides constant control over changes to managed information. An engineer made the comment:

"Why is this necessary? After all, as an engineer, aren't I most qualified to know how to fix a problem? I've got schedules and technical obligations to meet. I do not have the time to deal with this bureaucratic delay. I've got important

WHY CM IS ESSENTIAL (cont.)

work to do. The change is just five lines of code that I can patch. If I just change this interface slightly, the program will run twice as fast."

This represents the frustration of dedicated engineers forced to slow down because of a requirement to have proposed changes approved before they may be implemented. However, this discipline is essential if the integrity of the project environment is to be retained. The individual engineer cannot know the impact on testing, documentation, or agreed-to user and support interfaces that may be experienced if that change is applied to an operational configuration. The software engineer cannot anticipate the system impacts of just slightly changing an interface or the operational implications of a system failure caused by a patch that is incomplete or which has not followed established quality assurance procedures. Although appearing to be a burden, change control is the key discipline that ties the project together as information is modified to reflect project realities. This can be summarized as:

The Fundamental Law of Configuration Management

Configuration Management is the foundation of a software project. Without it, no matter how talented the staff, how large the budget, how robust the development and test processes, or how technically superior the development tools, project discipline will collapse and success will be left to chance. Do Configuration Management right, or forget about improving your development process.

PRIMARY CM ACTIVITIES

CM applies technical and administrative direction and surveillance to:

- Identify and document the functional and physical characteristics of a system, software, hardware, or operational component so that these relationships may be managed, maintained, controlled, and assured
- Control changes to those characteristics
- Record and report the status of proposed changes, and the status of the implementation of approved changes
- Disseminate baseline information to the project personnel, and establish and maintain a status accounting and reporting system that records the baseline, authorized changes to the baseline, and verification of changes incorporated into the documentation and/or product
- Review and audit the CM process to assure the process and the adequacy of control
- Establish a specific hierarchy of information for both nondeliverables and deliverables
- Establish CM metrics, and trend and alert indicators to support quality evolution of the software product

SETTING UP CM

Setting up CM requires an orderly planned sequence of activities and tasks. The following are key elements to establishing a successful CM program:

1. Plan and document the software CM requirements, activities, and responsibilities.
2. Establish a generic software development process and identify points at which quality gates will assess product integrity. Provide uniform format and content of engineering information, documentation, and review requirements.
3. Survey and select tools that can, without major modification, support the needs of the software process.
4. Establish the configuration identification scheme to be used for all software work products placed under configuration control. Configuration identification activities include the determination of the product structure, selection of Configuration Items (CIs), documenting the CIs, and allocating identification characters or numbers to the CIs and their documents.
5. Document and implement a method for collecting, recording, processing, and maintaining data necessary to provide configuration status accounting information. The process or procedure should include:
   - Identification of current approved configuration documentation and configuration identifiers associated with each artifact/work product
   - Status of proposed engineering changes from initiation to implementation
   - Traceability of changes for baselined documentation for each CI
   - Effectivity and installation status of configuration changes to all CIs at all locations
   - Defect identification and reporting from the CM problem reporting/corrective action system
   - Methods of access to information in configuration status accounting systems and frequency of reporting and distribution, including real-time, on-line access by all team members
6. Provide a framework for assessing the integrity of the development process and sharing information, and establish a means to integrate, qualify, and accept diverse components of a system.
7. Provide discrete points at which the schedule, budget, and quality of development may be evaluated. These are key items integral to the Project Control Panel.
8. Establish and implement configuration control procedures, designating: the level of control each

SETTING UP CM (cont.)

work product must pass through; those with authority to authorize or make changes at each level; and the steps to be followed to obtain authorization for changes, to process change requests, to track changes, to distribute changes, and to maintain past versions. Change control activities should be documented and be used to:

- Establish a Software Development Library (SDL) and change control process to establish baselines and to maintain the integrity and traceability of the configuration.
- Manage program baselines and ensure proper authorization of changes to those items selected for both internal and formal release from the SDL.
- Release and submit configuration documentation in relation to program events (i.e., technical reviews and design reviews).
- Establish internal development configuration and contractual baselines.
- Establish implementation and timing of internal and customer configuration control.
- Define the roles and responsibilities of the configuration control boards.
- Establish a problem reporting/change control process that is closed-looped to ensure that all detected problems are reported and entered into the system, action is initiated on them, resolution is achieved, status is tracked, and records of the problems are maintained for the life of the project.
- Establish a Change Control Board/Engineering Review Board with the authority for managing the project's software baselines. Products from the SDL are created, and their release is controlled through the CCB/ERB according to a documented procedure.
- Document the procedure/process, including roles and responsibilities, to be followed to request authorization for changes, process change requests, track changes, distribute changes, and maintain past versions. Changes that affect an entity already under configuration control will be proposed to the affected organizations (i.e., customer/subcontractor) in accordance with contractually established forms and procedures, as applicable.

9. Establish and document the configuration identification scheme to be used for all artifacts/work products placed under configuration control. The identification scheme should include the following types of items to identify artifacts/work products:
   - Both developmental and formal configuration baselines, including documents, SDLs, and corrective action process. The identification

SETTING UP CM (cont.)

scheme will be at the level at which entities such as computer files, electronic media, documents, software units, Configuration Items (CIs) will actually be controlled.

- Configuration identification activities, including determining the product structure, selecting CIs, documenting the CIs, and allocating identification characters or numbers to the CIs and their documents. The selection of work products placed under CM includes the software products that are delivered to the customer (for example, the code/build scripts, software requirements, software design, and software test procedures and documents) and the items that are identified with or required to create these software products (for example, the compiler, vendor-supplied/government-furnished information, and so on).
- Configuration identifiers, including document numbers and software identifiers for both software and firmware. The identification scheme should include the version/revision/release status of each entity.

10. Through configuration auditing, establish and document the plans and procedures for periodically auditing and evaluating the software products. In addition to specific audit requirements, a configuration evaluation/audit should be conducted to determine:
    - If the quality of the product meets both functional and physical requirements
    - If the work products are stored in a controlled library corresponding to the CM records
    - If the work products and baseline CIs are complete and available with respect to the status of the work products and approved modifications from which they are built
11. Establish and implement procedures for the packaging, storage, handling, backup, restore/recovery, and delivery of deliverable software products.
12. Establish sound subcontractor monitoring and auditing. Identification and re-creatability of artifacts or work products are required prior to acceptance from subcontractors.

Best Practice #1: Make Configuration Management Everyone's Job

From the inception of the project, create an environment where adherence to the established CM process is the only acceptable way of doing business.

Software engineering applications must be developed and maintained in an environment that enables communication and ensures a smooth, controlled flow of information and insight to development changes that affect the product. Many different organizations, often distributed across rooms, buildings, sites, or even continents, must work together to satisfy a requirement within a predetermined cost and schedule. Experimentation, rework due to poor control of information, and uncontrolled change cannot be allowed if the project is to succeed.

Symptoms of inadequate CM are not always obvious; they are often difficult to address and easy to misdiagnose. Problems such as nontraceability, inability to re-create a software product, interfaces and operational requirements that are poorly satisfied, or components that will not integrate because of interface inconsistencies are often blamed on technical rather than information management problems.

In many instances, catastrophic engineering problems occur because of inadequate information management and control—the problem solved by CM. The CM disciplines are prime risk reducers in a project environment. CM is vital to the success of any software effort.

Best Practice #2: Create an Environment and Engineering Process That Enables Configuration Management

Make every aspect of your development process help your people adhere to the CM discipline. Define the engineering/CM process first, and refine the process as necessary throughout the project's life cycle. Find those areas that cause developers and testers difficulty in maintaining CM, and make changes.

In examining the software development environment, the task is not simply to find ways to make it easier for software engineers to design and write code or testers to run tests. It is to apply the right controls at each phase of the development, so that baseline control is maintained and change control is airtight.

The CM planning structure should define an integrated CM environment. The software CM planning activities are hierarchical; requirements branch downward from a set of system development and control requirements. From these requirements, a software engineering structure is defined that satisfies the program goals and objectives and is consistent with system program constraints, limitations, and essential program relationships.

At the lowest level of the software engineering planning process, the requirements for software CM and specific procedures for managing and controlling the process are documented. The CM planning and process should be defined during the first stages of a proposal effort. This relationship is critical to the success of the software project and the quality of the products that are produced. If software is produced in a program environment that is unclear, inconsistent, or ineffective, the quality of the software will be questionable despite the rigor of the software development and the effectiveness of the software engineering process.

BEST PRACTICE #3

Best Practice #3: Define and Document the CM/Engineering Process, Then Select the Tool Set (Automation) to Support the Process.

CM and engineering must define the process, then the tools must be selected to support it.

The tool set, including the automated and manual components, must produce the products needed for baseline and change control, and auditing, without off-line work. The engineering CM and organizations must use the output of the CM tool set and must not require re-entering or manipulating data to perform required management activities. If the tool set cannot support this, the project will quickly become buried in data.

BEST PRACTICE #4

Best Practice #4: The CM Staffing Should Consist of Individuals with Technical Expertise to Support the Development and Maintenance of the Product.

When staffing the CM area, be sure the individuals have the technical expertise to develop and implement sound processes and procedures.

CM should be staffed with technically skilled individuals who can develop methodologies for control, re-creation and release of models, product build/compile structures, subcontractor monitoring and managing, and authoring of CM-related documentation (including, CM Plan, Software Product Specifications, Software Version Description (SVD) information, and so on.)

Be cautious when evaluating the staffing required to administer the CM system. While a mature system doesn't require a large staff, the project should not cut CM staff simply because things appear to be going well. The CM process must be institutionalized, not just written down. The tool set must be complete, in use, and no longer in need of constant maintenance. Several milestones—major deliveries to system integration, large test events, or audits—all become successful when supported by a fully utilized and reliable CM process.

Best Practice #5: The CM Plan and Procedures Need to Be Developed and Documented in the Same Way a Software Development Plan Is Created at the Initial Stages of a Project.

CM planning begins during the initial stages of a project. Staffing, budget, schedule, process, and procedures should be planned and included in the overall project's strategy.

The following are items to be included in a CM Plan:

- ***Introduction***—Sets down the purpose, scope, and applicability of the CM Plan to the overall project.
- ***Organization***—Describes the relationship and integration of CM with all organizations on the project (including, Quality, Engineering, Customer, Management, and so on.)
- ***Project Phases and Milestones***—Describes the evolution of the project with regard to CM and the establishment of both developmental and formal baselines.
- ***Configuration Identification***—Discusses the selection of artifacts or work products, configuration identifiers, establishment of the project libraries, and naming or numbering conventions.
- ***Interface Management***—Describes the procedures for identification of interface requirements, establishment of both internal and external interface agreement processes and procedures, and how the project will participate in the interface control working groups.
- ***Status Accounting***—Sets down how and when status accounting information will be recorded and released for the project. Information contained in status accounting includes the current release of work products and status of change.
- ***Configuration Audits***—Describes the plans and procedures for performing audits and verification of a product.
- ***Subcontractor Management***—Describes the process and procedures to be used to ensure the subcontractors' compliance with project requirements.

CM RULES

RULE 1: CM must manage the ownership of information and bring work products under control in a timely manner.

RULE 2: Early identification and change control of artifacts and work products is integral to a project.

RULE 3: The change management process must be simple, consistent with the culture and requirements of the project, and supported by the methods and tools of the project.

RULE 4: The actions, roles, and responsibilities of the Engineering Review Board (ERB) and the Change Control Board (CCB) should be documented.

RULE 5: Change control must be a primary focus of the program and must be integrated into the culture of the project.

RULE 6: All information that is placed under CM control or promoted in the CM hierarchy must be promoted based on successful completion of a quality gate.

RULE 7: All proposed changes to CM-controlled products should be properly classified.

RULE 8: All CM library structures or releases from the CM library must be recorded in a Software Version Description (SVD) document or comparable document.

RULE 9: It is essential that CM be continually aware of the exact status of products under control, the relationships between these products, problems, and open issues, and who is using the products.

RULE 10: The worst way to establish a CM system is to purchase a tool set and then devise a process to fit the tools. Establish the CM process that fits your project and culture, then find a tool set that supports that process.

COROLLARY 1: Artifacts used for evaluating CM effectiveness must be products of the system. If the CM tool set (manual, automated, or some combination of both) does not produce the records necessary to verify CM integrity, the tool set is flawed.

COROLLARY 2: Only the most trivial development can survive without automated tools.

COROLLARY 3: A mature, stable CM system (process, procedures, and supporting tool set) does not require an army to administer it. A few good staff members should be able to run any development.

RULE 1: *CM Must Manage the Ownership of Information and Bring Work Products under Control in a Timely Manner.*

One critical aspect for control of work products is the proper timing for when they enter into CM. The timing for entering into CM should be synchronized with the project phases. The following table provides an example of the types of project phases, including the input, output, and resulting type of baseline.

REVIEW/AUDIT	INPUT	OUTPUT	RESULTS/BASELINE
System Requirements Review	Updated System Specification Preliminary performance budget allocations	Final System Specifications Traceability of system requirements Performance/error budgets	Functional Baseline
System Design Review (SDR) or Preliminary Design Review (PDR)	Software Requirements Specifications (SRSs) Interface Requirement Specifications (IRSs) SRSs/IRSs traceability	SRSs/IRSs update Update of requirements traceability to SRSs/IRSs and Software Development Description budgets	Allocated Baseline
Critical Design Review (CDR)	Detailed design Detailed interface design Test procedures CI test plan	Detailed design updates Updated interface definitions Updated traceability	Preliminary Product Baseline or Detail Design Baseline
Test Readiness Review (TRR)/Site Acceptance Test	Establishment of configuration baseline, test documentation adequacy, compatibility with requirements, demonstration of performance in relation to system requirements	Detailed design approval to start testing Baselined software and test equipment Traceability of tests to requirements	System Performance Test Baselinet
Configuration Audit	Verification of all CIs against system documentation	Configuration Controller Product Baseline	Product Baseline

Figure 2. Milestone and Baseline Timing

For CM to work efficiently and effectively, clear responsibility for information ownership must be defined upon the level of maturity and timing of control as defined in the project's CM Plan. The assignment of this responsibility should be used to partition the control of information. An example of how ownership is defined is shown in the figure below:

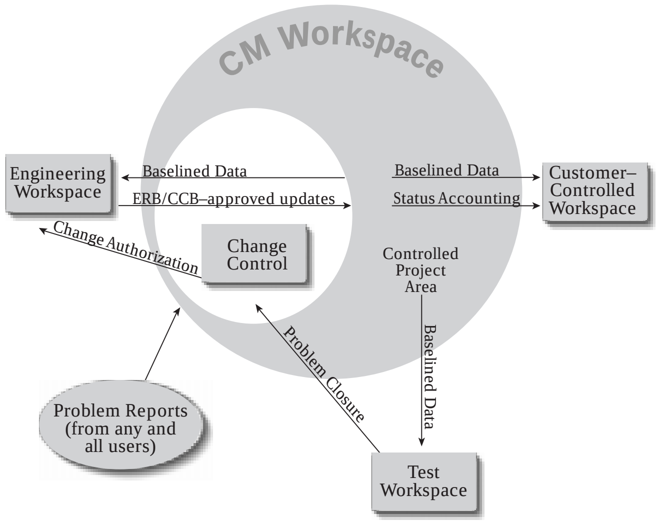


Figure 3. Configuration Control Flow

RULE 1 (cont.)

Configuration Control of Information

- Engineering Workspace
  - Holds all information that is under development or being changed, that has not been approved for program use or release
  - The information may be changed by the individual responsible for the product as the approving agent
  - The content is controlled by the individual responsible for the information
  - Workspace is owned by the engineer doing the work
- CM Workspace
  - Stages of documentation and software re-creating (compiling) and releases
  - Holds tools and files used by the project, or needed to build or control project information and/or deliverables
    - Project plans, procedures, and reports
    - Schedules, budgets, and CM records
    - Audit trails, records, and lessons-learned information essential for certification, approval, or acceptance
  - Workspace is owned by CM, with read-only privileges to all other organizations
- Controlled Project Area
  - Holds all information or documentation that has been accepted for release inside the program but has not yet been accepted by the customer or owner of the information at a higher-level life cycle for inclusion in a baseline
  - Includes information approved for internal release through passing of a quality gate
  - Requires that the program establish a process for CM of evolving information that is shared within the project by different organizations
  - This information cannot be updated unless the change is authorized by an established project board
  - Project area is owned by CM, with read-only privileges to all other organizations
- Test Area
  - Holds all test documentation and versions of software released for test from CM
    - Test scenarios, data, and results
    - Test configurations
    - Test cases
    - Software configurations under test and configuration documentation
    - Special test data, records, and audit trails
  - Test area is owned by test manager

RULE 1 (cont.)

- Customer-Controlled Workspace
  - Holds all customer-approved baselined and approved information
    - Functional
    - Allocated
    - Product
    - Records and reports
    - Other approved or delivered information
  - This area cannot be updated unless the change is authorized by the customer
  - Workspace is owned by the customer

RULE 2

RULE 2: ***Early Identification and Change Control of Artifacts and Work Products Is Integral to a Project.***

The project needs to fully identify and control changes to all work products required to re-create and maintain the software product. Work products encompass more than just deliverables. All support tools, test drivers, Commercial Off-the-Shelf (COTS), and development environment items should be considered. The following provides a list of the types of artifacts and work products to consider:

- Plans
  - Systems Engineering Management Plan
  - Software Development Plan
  - Software Standards and Procedures Manual
  - Software Configuration Management Plan
- System Specification
- Software Requirements Specification
  - Graphical analysis modes
  - Process specification
  - Interface specification
  - Prototype(s)
  - Mathematical specification

RULE 2 (cont.)

- Design Specification
  - Data design description
  - Architectural design description
  - Module design description
  - Interface design description
  - Software design folder
  - Objects description (object-oriented)
- Source Code
  - Source code
  - Build scripts
  - Source code listings
  - Executable programs
  - Module executable code
  - Linked modules
  - Legacy code (i.e., COTS, Government-Furnished Information (GFI), Non-Development Item (NDI), and so forth)
- Operation and Installation Manuals
- Test Specification
  - Test plan and procedure
  - Test cases and recorded results
- Database Description
  - Schema and file structure
  - Initial content
- Software Description Document/Version Description Document
  - Compile and build instructions
  - Compiler(s) or network environment
- COTS products
  - Compilers
  - Computer-Aided Software Engineering (CASE) products
  - Models
  - Simulation software

RULE 3: *The Change Management Process Must Be Simple, Consistent with the Culture and Requirements of the Project, and Supported by the Methods and Tools of the Project.*

- The process starts with a program stakeholder, engineer, user, or any other individual identifying, documenting, and prioritizing a problem and then assessing, if possible, the impact of the problem and the recommended correction, cost, and schedule impact of the change, essential resources to correct, and the effects if not corrected.
- The CM process must be adapted to, and be consistent with, the cultural aspects and realities of the project, or it will prove ineffective and will be resisted by the project staff.
- CM must be a centralized activity integrated into the engineering process, but it should remain independent from both engineering and quality assurance organizations.
- CM activities must be planned and tightly coupled to the test schedule. The work products released from CM become the lifeline for the testing of a software product.
- CM can never be bypassed for any reason, including the need to recover budget or schedule.
- Suspected and confirmed problems must be reported through CM change control, and impacts must be assessed prior to authorizing a change.
- Every suspected or observed problem or program issue, no matter how small, must be processed through the problem reporting/corrective action system.
- All trace information between project documentation, deliverable documentation and engineering artifacts, and test cases and the information being tested must be CM-controlled information that has been approved through a quality gate.
- The proposed Software Problem Report (SPR) is reviewed by the submitters' or proper Engineering Review Board/Change Control Board (ERB/CCB) and is either sent back for revision or approved for release to the CM library. The CCB/ERB evaluates each fix, assessing the process used to make it against the documented project process, program plans and constraints, and assurance requirements and standards; and then assesses whether the fix is consistent with the standards for the product as required by the customer.
- All formal test builds must be managed by, controlled by, and the responsibility of CM.
- All releases from CM must be accompanied by a current, complete, and accurate Software Version Description (SVD) document or comparable document.

RULE 3 (cont.)

- All product releases should be regression-tested prior to deployment or release to the customer.
- Engineering changes that do not follow the CM process cannot be used by, or shared within, the software project.

RULE 4

RULE 4: ***The Actions, Roles, and Responsibilities of the Engineering Review Board (ERB) and the Change Control Board (CCB) Should Be Documented.***

Actions that may result from the ERB/CCB are:

- ***Reject***—Reject the Software Problem Report (SPR).
- ***Assign for Study***—Assign the SPR to the technical organization for analysis.
- ***Assign for Correction***—Assign the SPR for correction, authorizing update.
- ***Append to Open SPR***—Block the SPR with another problem under analysis.
- ***Reanalysis***—Send the SPR back for further analysis.
- ***Table***—Defer analysis, but don't redefine suspense date.
- ***Defer***—Defer analysis and assign suspense date.
- ***Identify Testing and Retest Requirements.***
- ***Build System and Release***—Authorize a software build and release.
- ***Close***—Close the SPR. Problem is resolved.

ERB / CCB Responsibilities

General ERB/CCB Responsibilities

The primary functions of the ERB/CCB are to:

- Ensure that proposed changes to hardware, software, and documentation are systematically evaluated with respect to their impact on other related program elements
- Review and approve/disapprove all proposed changes to controlled hardware, software, and documentation
- Ensure that only necessary changes are approved
- Ensure that only approved changes are implemented
- Ensure that proposed changes are properly classified
- Determine urgency for incorporating the changes and establish effectivity points

Chairperson's Responsibilities

The chairperson is responsible for resolving disputes regarding proposed changes and for implementing the ERB/CCB aspects of the CM Plan. As required, duties include:

- Requesting and evaluating an impact analysis
- Making a decision on changes
- Giving final approval on all changes within his/her jurisdiction
- Keeping higher management advised of significant changes in design, cost, and/or schedules
- Establishing criteria for the acceptability of proposed changes
- Determining the needed reviewers of a change
- Assigning action items, as required

ERB/CCB Reviewer Responsibilities

- Each reviewer of a change should obtain all the information needed to assess any impact of the change.
- Each reviewer should be prepared to negotiate for the component represented and have authority to commit the organization to take action on changes being reviewed.
- Each reviewer should complete an impact analysis detailing the impact to the organization he/she is representing.
    - This impact analysis should be performed within the review time allocated by the ERB/CCB.

RULE 5

RULE 5: ***Change Control Must Be a Primary Focus of the Program and Must Be Integrated into the Culture of the Project.***

- ***The project has to have a mechanism to introduce change:*** Change is very often a function of time—the longer the duration of the program, the more change. There must be a controlled, disciplined process for presenting changes to the program community.
- ***Establish criteria by which to classify changes:*** Not all changes are equal, and there must be a method to determine importance and priority to allocate resources.
- ***Consistency across products:*** The impact of a change must be assessed by all disciplines within a program. All products must be appropriately modified to reflect the change.
- ***Timely and comprehensive visibility of change status:*** Timely and comprehensive visibility of all changes is needed through the ERB or CCB.
- ***A historical record of all changes:*** This record, or audit trail, should be consistent with the tracking, reporting, and certification requirements of the program.
- ***Changes must be adjudicated before they are made:*** The project should have a process to evaluate project input and a forum through which to adjudicate the implementation of a change before it is made.

RULE 6

RULE 6: ***All Information That Is Placed under CM Control or Promoted in the CM Hierarchy Must Be Promoted Based on Successful Completion of a Quality Gate.***

- All products controlled by the program must be built in accordance with established project standards and standards of workmanship unique to the product requirements, essential characteristics, and operational needs.
- Quality gates used to approve information for increasing levels of CM control must be structured inspections, walkthroughs, successful completion of planned testing activities, project reviews, or independent product analyses based on the characteristics, contractual status, use, and risk associated with the product.
- For each quality gate conducted, a threshold below which the specific product will not be approved must be defined. These thresholds serve as quality targets, by product, that must be met by the product.
- Product standards must identify what the documentation, engineering products, and hardware and software products look like, what they should contain, and how they should be structured, tied together, and packaged for release.
- Engineering Review Boards and Change Control Boards play a critical role in technical and management reviews of proposed changes and in

the implementation of changes. It is essential that these boards continually ensure that what is under control is actually what was placed under CM and approved through a quality gate.

- CM must monitor library content, assessing release level, content, applied changes, and the approval level of controlled information.
- CM must ensure that all products approved for control have required approvals and that quality assurance has confirmed the products meet required standards.
- CM must ensure that change requests are processed correctly before a change to a controlled library is made.
- CM must audit sites that have received a release to ensure that the approved and released configuration is used.

RULE 7: ***All Proposed Changes to CM-Controlled Products Should Be Properly Classified.***

Classes of changes to CM-controlled products include:

- ***Class I Changes***—Changes that affect the functionality or performance requirements, such as form, fit, function, or contractual cost, schedule, delivery, contract guarantees or warranties, or other contractually agreed-to factors. This includes memoranda of agreement or agreed-to work packages between in-house organizations.
- ***Class II Changes***—Minor documentation changes that do not affect factors associated with Class I.
- ***External Changes***—Changes to jointly owned or managed information (for example, interfaces), where the organizations owning the information do not have a contractual relationship.
- ***Internal Changes***—Changes that affect information owned or used by someone else within an organization, but which have not been approved or delivered to the acquirer. These changes must be capable of implementation and must be within budget and schedule constraints.

RULE 8

RULE 9

RULE 8: ***All CM Library Structures or Releases from the CM Library Must Be Recorded in a Software Version Description (SVD) or Comparable Document.***

- The SVD documents the CM library structure stock list by product version ID; open problems; product included in the structure; closed problems resolved by this product release; differences between this version of the CM structure and previous release; and notes, limitations, assumptions, and any other critical information related to the partition.
- The SVD should include the environment necessary to maintain or re-create the product: compiler name and version, operating system and version, command or build files, and compile instructions.
- All information released from CM, whether it be documentation, software, or full customer releases, should be documented through an SVD provided as part of the release.
- The SVD should be comprehensive, documenting specific content to the lowest-level controlled item, and should be built and maintained as part of the automated CM tool.
- SVDs should be available electronically to all users and stakeholders associated with the information.
- All individuals affecting or affected by a release should be notified of the availability of the new SVD.

RULE 9: ***It Is Essential That CM Be Continually Aware of the Exact Status of Products under Control, the Relationships between these Products, Problems, and Open Issues, and Who Is Using the Products.***

- All library structures should contain a current index of the exact content of all products, version ID, release status, and who is using the products. The index should be hierarchical, treeing down to the lowest-controlled product.
- Each library structure should include all traceability matrices that identify relationships between products and the products and activities that assure them.
- The CM library should maintain a cross-reference of open SPRs and affected products.
- The CM library should maintain records of who has checked products out for change, and should lock the product against change until it is checked back in.

RULE 10: ***The Worst Way to Establish a CM System Is to Purchase a Tool Set and then Devise a Process to Fit the Tools. Establish the CM Process That Fits Your Project and Culture, then Find a Tool Set That Supports That Process.***

Questions to ask when selecting CM methods or tools:

1. Does a process exist for CM from which the requirements for an automated tool may be derived?
2. Have the proposed CM techniques or tools been proven in applications of similar size and complexity with similar environments and characteristics?
3. Are the proposed CM techniques or tools consistent with the documentation and review requirements of the customer or contract, and can the products that result be used without modification in other phases of development?
4. Is there clear traceability between requirements specified by the CM process or tools and those needed by related methodologies or tools? This is essential since the CM tool is the basic integrator of information and, as such, must link to all components of the project.
5. Is it possible to plan the overall information and organizational flow of the project, interfaces, and relationships between project segments and to define a schedule that integrates specific requirements of the selected methodologies into a consistent and effective project structure for test?

Corollary 1: ***Artifacts Used for Evaluating CM Effectiveness Must Be Products of the System. If the CM Tool Set (Manual, Automated, or Some Combination of Both) Does Not Produce the Records Necessary to Verify CM Integrity, the Tool Set Is Flawed.***

Corollary 2: ***Only the Most Trivial Development Can Survive without Automated Tools.***

Corollary 3: ***A Mature, Stable CM System (Process and Supporting Tool Set) Does Not Require an Army to Administer It. A Few Good People Should Be Able to Run Any Development.***

- Monitor the output of the CM process—the reports, reviews, and audits created or carried out in the course of day-to-day business. If no one reads or uses the reports, eliminate them. If the reviews and audits don't produce measurable improvement in the product, stop performing them.

INDICATORS OF CONFIGURATION CONTROL EFFECTIVENESS

Answers to the following questions provide indicators of the effectiveness of a program's configuration control process. The indicators provide a "quick look" at how well configuration control supports program requirements.

1. Configuration Identification Process:
   a. Does the program have in place a documented process for uniquely identifying each baselined document or approved management, engineering assurance, or reporting product, and are these controlled items current (at the most recent release level)?
   b. Has the program partitioned this information into Functional, Allocated, Design, and Product Baselines, and does it maintain traceability and control among them?
   c. Is there a documented and approved process in place to accept information into the configuration control process and to update it once it's been baselined?
2. Change Control Process:
   a. Does the program have in place an approved process for proposing changes to baselined information and for evaluating and approving these changes prior to implementation?
   b. Does the program have in place a documented process for evaluating the management, technical, and assurance impacts of a change to baselined information, and are these impacts considered before a change is approved?
   c. Does the program maintain records and audit trails on the status of proposed changes that are pending or have been approved for implementation?
3. Status Accounting Process:
   a. Does the program have a documented and agreed-to understanding of the configuration control requirements for delivery, and is a process in place to implement it?
   b. Does the program have in place a means to monitor and document the status and currency of all information that it controls, and does it document this status when a release is made through the configuration control process?
   c. Does the program document all releases through a Version Description Document (VDD), and does this VDD contain a stock list of what is in the release by version, the problems closed by the release, problems still open with this release, and limitations and restrictions of the release?

INDICATORS OF CONFIGURATION CONTROL EFFECTIVENESS (cont.)

4. Configuration Control Review and Audit Process:
   a. Does the program conduct regular reviews and/or audits of information against approved standards, expected delivery content, and/or currency and completeness of baselines?
   b. Is the configuration control process an integral part of the program acceptance and release process, and is CM responsible for Physical and Functional Audits in support of this requirement?
   c. Can configuration control tell the status of all baselined information, and does the CM organization regularly conduct reviews or audits to verify the status of items under its control?
5. Configuration Control Planning:
   a. Does the approved configuration control plan accurately reflect the process being followed, and is it current and approved?
   b. Is the configuration control process adequately supported by effective tools, and are there sufficient resources and organizational support and discipline to support the requirement?
   c. Is there a process improvement plan in place to upgrade the configuration control process, and is this plan implemented and adequately supported?