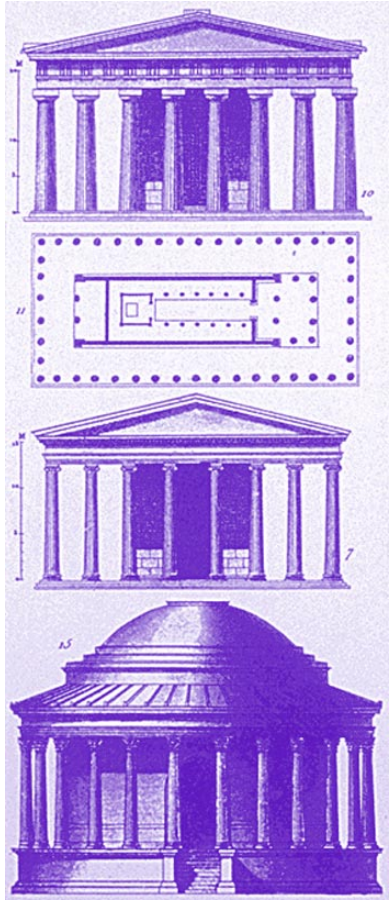
S T S C

Software Technology Support Center

Software Configuration Management Technologies and Applications



STSC Technology Report

Sponsored by

E S I P

MISSION CRITICAL SOFTWARE

Embedded Computer Resources (ECR) Support Improvement Program

Executive Summary

Software Configuration Management (SCM) is a discipline to manage the evolution of computer program products during all stages of development and sustainment. The benefits derived from SCM are directly proportional to the extent that SCM is implemented. An effective SCM program: (1) supports delivery of a product that meets the stated requirements, (2) tracks each requirement and deliverable from concept through implementation to customer delivery, and (3) ultimately moves toward delivery of the product on schedule and within budget.

Measurements can be collected and combined into one or more metrics, which identify the performance of the processes. They are used to measure the progress of a project and the quality of its product.

The importance of SCM becomes evident in light of the return on investment and cost savings potential. When SCM, measurements, and their associated metrics are used from the beginning of a program through its conclusion, the payoff can be enormous.

With each new software project or process, there is some amount of risk associated with it. Managing and controlling the risk is essential to the success of the project in terms of cost, schedule, and quality. An SCM risk management plan should be developed which focuses on three areas: *business or product, human resources , and technology.*

SCM encompasses the every day tasks within an organization, whether they be software development or sustainment. The software changes are identified, controlled, and managed throughout project lifecycle.

SCM is divided into the following functional areas: Identification, Change Control, Status Accounting, and Audit. An SCM plan describes what is to be done in each of these areas and "provides the focus for the process and procedures, and is the mechanism used to communicate the CM process to the other organizational groups on the project." [Bounds 96] Automating SCM consists of all the steps involved in introducing an SCM tool into an organization and ensuring that it is routinely used on all projects. Implementing a series of six phases can successfully carry out SCM automation. They are preparation and planning, process definition, tool evaluation, pilot project implementation, rollout to other projects, and process improvement. The phases provide structured guidance, identify tasks, and address the complexities involved with automating SCM.

Table of Contents

Abstract

Software Configuration Management (SCM) is the backbone of the software development process. When it is implemented correctly it helps ensure software quality and process improvement.

The purpose of this report is to provide current information on basic SCM principles, methods, and technologies, and identifies their value in improving software quality. It should be used as a first step in transferring effective SCM processes and products into practical use.

This report also provides an overview of SCM concepts, what it is and how to implement an SCM process as well as SCM standards, current trends and future directions for SCM, measurements and metrics, case studies, and lessons learned.

Emphasis is placed on the need for organizations to develop a long-term SCM solution by developing a detailed SCM plan and defining the SCM process before implementing an automated system.

Acknowledgements

This report was prepared by the following individuals: Janiece Jones, Paul Hewitt, Russell Lee, Larry Smith, and Reed Sorensen. Special thanks to Bruce Angstadt and Bob Ventimiglia for their valuable contributions.

Mr. Angstadt is an independent consultant in SCM with over 30 years experience in private industry programs and government agency funded contracts. He can be reached at: 269 Marion Street, Indian Harbour Beach, FL 32937; (voice) 407-777-7914; (fax) 407-723-9852; info@scmtoday.com. He also teaches SCM, Engineering Support Management, and related disciplines for System Technology Institute, P.O. Box 6907, Malibu, CA 90264-6907; www.stitraining.com.

Mr. Ventimiglia is a SCM Manager for Lockheed Martin Aeronautical Systems where he is leading advanced Effective SCM implementations for LMAS Software-Intensive Systems on the Hercules C130J and Spartan C27J programs. He can be reached at: Lockheed Martin Aeronautical Systems, 86 South Cobb Drive, Marietta, GQ 30063-0685; (voice) 770-494-9791; (fax) 770-494-1345; bventimi@hercii.mar.lmco.com. He also teaches SCM and Advanced SCM for Technology Training Corporation, P.O. Box 3608, Torrance, CA 90510-3608; www.ttcus.com.

For more information, please contact the STSC below.

Software Technology Support Center
United States Air Force
Ogden Air Logisitics Center (OO-ALC/TISE)
7278 4th Street
Hill Air Force Base, Utah 84056
www.stsc.hill.af.mil
Fax: 801-777-8069

Overview of Software Configuration Management

1

1.1 The Role of SCM

Software Configuration Management (SCM) is the discipline for managing the evolution of computer program products during all stages of development and sustainment. During the last decade, software technology has progressed at breathtaking speeds. However, our ability to manage the complexity of problems with software development and our ability to develop processes to handle this rapid change has not increased as quickly.

> The ability to develop and deliver reliable, usable software within budget and schedule commitments continues to elude most software organizations. After two decades of unfulfilled promises about productivity and quality gains from applying new software methodologies and technologies, organizations are now realizing that their fundamental problem is the inability to manage the software process, if in fact any organizational process exists. In many organizations, projects are often excessively late and over budget, and the benefits of better methods and tools cannot be realized in the maelstrom of an undisciplined, chaotic project. [Paulk 95]

SCM provides the means to manage software processes in a structured, orderly, and productive manner. SCM spans all areas of the software lifecycle and impacts all data and processes. Hence, maximum benefit is derived when SCM is viewed as an engineering discipline rather than an art.

Depending on where you go and who you ask, you will always get different answers to the question, "Is Software Configuration Management a support or a control organization?" Even if you ask two people working on the same program, you are likely to get different answers. Those people who are afraid that it's a trick question, or because they want to appear to be a bit on the profound side of things, will respond with, "Why, it's both, of course." But when asked what they mean by that, well, they usually have other pressing matters to attend.

The answer is that SCM is both a support and control organization, and when it is handled properly, a third factor is drawn into the picture, that of being a service.

If you talk with a software developer, they are likely to tell you that they want support and some measure of control, but not too much. Management on the other hand will generally say that controls are the more important issue, as long as they don't impact costs and schedules. An SCM person with 10 or more years of experience, probably a first line supervisor or manager, will agree with management but will also insist that sometimes cost and schedule must be impacted. An SCM person with fewer years under their belt will generally side with the software developer and strive to do whatever is necessary to assist them and get the job done.

The question becomes "How can SCM best accomplish both support and control issues and be of service while adding value to the program effort?"

Support. SCM is a support function in that it supports program engineers and developers, the program, the corporation, and in a number of situations, the customer.

Control. SCM is a control function in that it controls specifications, documents, drawings, requirements, tools, software, and other deliverables.

Service. SCM is a service provider in that it supports people and controls data. This one simple sentence is the primary key to a successful configuration management operation.

The SCM staff must be able to wear two completely different hats: one to support people, and one to control data. And when those two hats get mixed up, i.e., SCM tries to control people and what/how they do things, problems and bottlenecks show up on the immediate horizon. And when that happens, all too often SCM is bypassed for the sake of "get the job out the door and we'll fix it later."

Additionally, the role of the SCM Manager is to ensure that: (1) SCM personnel are properly trained and have the necessary resource (budget and tools) to do an efficient and effective job; (2) a proper balance of control and support is tailor made to each program that is being supported; and, (3) the SCM function is flexible and can accommodate the changing needs and requirements of the developers, customers, the program and the company.

1.2 State of SCM

The SCM task has not really changed much during the last 20 to 30 years. However, the environment that SCM operates within has changed significantly and is likely to continue to change. Mr. Angstadt adds:

Certainly the software language bases have changed; from Basic, COBAL and FORTRAN, to Ada and Pascal, to C++, Java, and numerous others. But that has not been the real impact to SCM; after all, code by any other name is still code.

The more significant impacts to SCM have been centered on the automated tools and the library systems they operate on.

The tools have progressed from version control and semi-automatic build operations to systems that can now establish and monitor the entire software development and production environment. The tools are more sophisticated and the suppliers more numerous. Not long ago it was a somewhat simple matter for SCM to determine the best tool for the job. But in today's market new issues have to be addressed before a decision is made. It has become increasingly important for representatives from each department within engineering organizations to now consider, evaluate, and weigh their requirements against the capabilities of the various tools that are available. The SCM automated tools available today, and those still on the drawing boards, are much more versatile than their forerunners. However, when asked, "Isn't there one tool out there that will do it all?" The answer is still no! And it is due in large part to the fact that the environment SCM operates within is still evolving.

1.3 Current Trends in SCM

In the not too distant past, SCM dealt with code and a few documents, each carefully tucked away into a baseline where it could be easily controlled. Mr. Angstadt further adds:

With the introduction of web based repositories and significantly increased involvement with Commercial-off-the-shelf (COTS), vendor, and subcontractor applications, baselines, as originally defined, are quickly becoming a thing of the past. Baselines are becoming conceptual in nature. After all, when is the last time you "saw" a Functional, Allocated, or Product Baseline? Probably when you were dealing with hardcopy documents and printed listings of programs. But in the electronic office these are no where to be found. The tendency now is to refer to the current version of controlled entities (code, documents, requirements, etc.) as the *Baselined Version.* Previously controlled versions are now referred to as the *Archived Baseline Version*.

In the past, most programs operated with three baselines: Functional, Allocated, and Product, or Requirements, Design, and Product, or some other set of descriptive labels. The National Aeronautics and Space Administration (NASA) once tried a system with nine baselines; it had a short life span. NASA, and the greater majority of other developers as well, were going on the assumption that it was important to know into which baseline the controlled items were placed. Actually, the primary concern should be that the item is baselined and not changed without going through a formal process; not simply a name given to an electronic partition where the item resides.

What is driving this move away from our three traditional baselines? It is the questions being raised by SCM and other developers. Questions like "Where do I put this .JPG file that is used in different documents?", "Shouldn't the *Build Scripts* and *Make Files* be controlled too?", "Where do we control corporate assets known as, or to be captured as, Re-usability or Re-engineered issues?", et cetera, used to be asked.

The questions, however, are now becoming: "Does the control board control the document or the information it conveys?", "Does the control board control the software code or what it does, how it does it, and all things used to create it?", et cetera.

In the past, SCM controlled code and sometimes, but not always, documentation. What can be baselined in the environments we are beginning to deal with now? The easier question is "What can't be baselined?" Answer, SCM can baseline anything that the program needs to control and make available. Answer, SCM controls data (in any form) so that it can support people and provide an integral service to the program.

Importance of Software Configuration Management

2

2.1 SCM and Process Improvement

The theme of the 1998 Software Engineering Institute Symposium was "Improving WHAT you build means improving HOW you build" [Carnegie 98]. In other words, improvement depends on changing current processes and the accompanying environment. According to an old adage "if you do what you've always done, you'll get what you've always got." SCM provides the underlying structure for change and process improvement.

For example, the first step to improve the product is to know how the product is currently produced. The second step for improvement is to foster an atmosphere where change can be accommodated. If change does not appear possible, then improvement is also unlikely. SCM measurements of current practices and their associated metrics can help identify where processes are working and where they need to be improved. Such change efforts should lead to increased productivity, integrity, conformance, and customer satisfaction. A change must add value, or it should not be made.

The Institute of Configuration Management (ICM) defines configuration management (CM) as "the process of managing the full spectrum of an organization's products, facilities and processes by managing all requirements, including changes, and assuring that the results conform to those requirements" [ICM 98]. By this definition CM can also be called *process configuration management*, since it includes the process of managing an organization's processes and procedures.

Many organizations can be characterized as Level 1 organizations as defined in the Software Engineering Institute's Software Capability Maturity Model (SEI SW-CMM). These Level 1 organizations rely heavily on "heroes" to accomplish the work. The organization's processes are not documented and few people know how the work is accomplished. "The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics." [Paulk 95]

An effective SCM program, when applied to organizational processes, identifies which processes need to be documented. Any changes to those processes are also tracked and documented. Adhering to these processes will reduce an organization's dependence on heroics for the work to be accomplished and the project to succeed. It also relieves the frustration and problems that arise if one of the "heroes" is not available to perform a task.

SCM is an essential discipline in the every-day activities of defining requirements, designing, writing, compiling, testing, and documenting the software. SCM is not simply version control or format control. It is not a clerical "after-the-fact" function. It is a technical field of expertise with formal practices.

The benefits derived from SCM are directly proportional to the extent SCM is implemented. The primary objective is to deliver a quality product that meets the stated requirements, on schedule, and within budget. An effective SCM program supports this objective by tracking each requirement from concept through implementation to customer delivery.

2.2 Measurements and Metrics

The status accounting aspect of SCM provides management visibility into the state of the software products. Status accounting data includes measurements that can show the location of bottlenecks in the software development and sustainment process, and can indicate the maturity of the software products.

As an example of the later of these two, extracts from [Hermann 98] describe the use of software changes to measure product maturity and readiness to deliver the software (*see side bar on this page and the following*). This articles goes on to mention other metrics that may be useful, including: average severity, severity level distribution, average closure time, charts for each severity level, and charts for each configuration item or subsystem.

Beyond the scope of the aforementioned article, a brief discussion on measures and metrics, as they pertain SCM, can assist SCM practitioners in their efforts to manage and control software systems.

A measure can be defined as "a standard of measurement, the extent, dimensions, capacity, etc., of anything, especially as determined by a standard, an act or process of

External factors

To correctly gauge readiness to deliver, developers must also evaluate test completeness, test rates, and requirements stability.Any of these factors can cause product maturity to look unrealistically good or bad. Obviously, if only 10 percent of the planned tests have been completed, it is premature to ship the product—despite low software change trends. Likewise, high test rates will likely produce more change and problems than lower test rates. Requirements instability is one of the most common causes of software product immaturity of the Department of Defense's long development cycle projects.

Trend Charts and Analysis

Software product maturity evaluation entails a graphics analysis of change data trends in the context of project schedule and other external factors. The basic product maturity chart (Figure 1.) shows the *total changes originated*, *closed*, and *remaining* trends. (Note: These charts contain data from

Software Product Maturity Data Requirements

Our experience shows that most developer and procurement offices already collect the data necessary to perform a software product maturity evaluation (and consequently readiness to deliver). Data that describes and tracks documented software changes served as the key input to the evaluation. Following are the minimum data required for each software change to evaluate software product maturity.

- Software change (problem) number
- Description
- Computer Software Configuration Item (CSCI) Identifier.
- Severity level.
- Date change opened (or problem found)
- Date change (problem) closed and implemented.

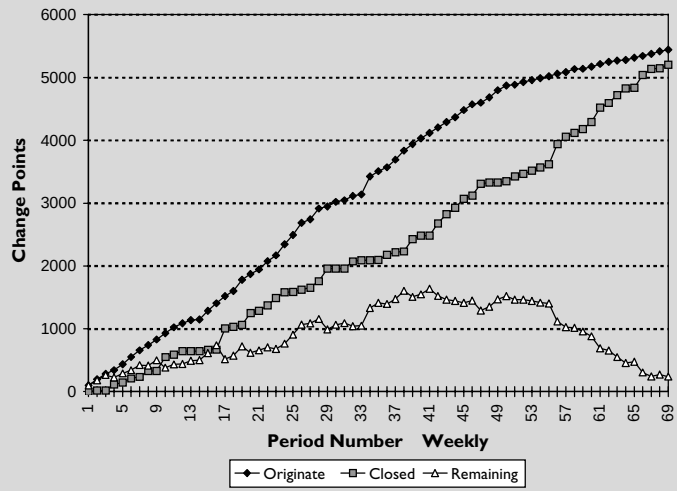


Figure 1. Accumulated software changes (weighted).

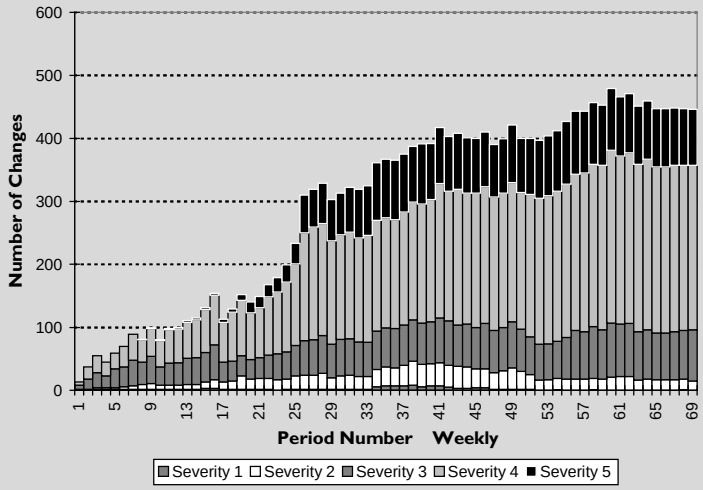


Figure 2. Remaining software problems (unweighted).

multiple, real systems and are provided as examples only.) To indicate maturity or progress toward maturity, the total changes originated trend should begin to level off. This indicates testing is finding problems at a decreasing rate. If problems are being closed efficiently, the total changes closed curve should closely follow the total originated trend. Ideally, all identified changes are closed, and the remaining changes curve would show no backlog.

Although the *remaining changes* trend in the basic chart shows the current software problem or change backlog, Figure 2 presents a more useful view. This stacked bar chart shows the overall backlog trend as well as each severity level's contribution to the total backlog. (Note that the severity levels are those found in MIL-STD-498 in Appendix C, where 1 represents mission threatening and 5 represents less impact than a user inconvenience.)

Figure 3 shows both remaining changes for each CSCI sand the defect density (the number of remaining changes or problems divided by thousands or new or modified source lines of code.) In addition to the minimum change data, defect density analysis requires code size information. Literature suggests software is not ready for release until the defect density is below 0.5 [1]. Rather than blindly endorse this number, we suggest developers select a threshold of their own. Finding portions of software with the most remaining problems and the highest defect densities are two additional pieces to the product maturity puzzle.

Reference

1. Foody, Michael A., "When is Software Ready for Release?" *UNIX Review,* March 1995.

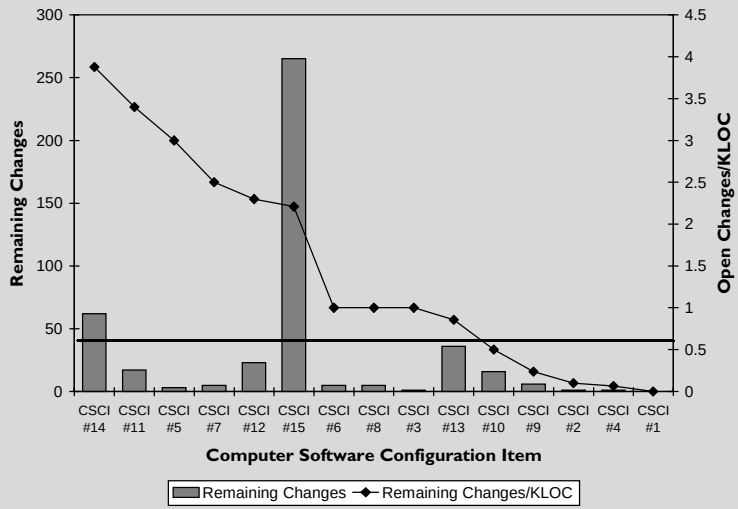


Figure 3. Remaining changes and defect density.

measuring, a result of measurement" [Starrett 98]. Examples of a measure are the number of defects found in a release or the number of source lines of code delivered.

A metric can be defined as "a calculated or composite indicator based on two or more measures, or a quantified measure of the degree to which a system, component, or process possesses a given attribute. An example of a metric is defects per thousand source lines of code" [Starrett 98].

A metric can also be "a composite of measures that yields systematic insight into the state of processes or products and drives appropriate actions" [Pitts 97]. Measures (measurements) and metrics can be used to identify areas of the process that require attention. These areas are identified through compiling measurements into metrics. Measurements are compiled in an electronic spreadsheet, database, or by hand. There are also several management tools that allow collection of measurements and derivation of metrics. The format is not the issue, the data is. Examples of some basic measurements are listed on the following page in Table 1.0.

A metrics program should include the following fundamentals [Pitts 97]

- A motive which is compelling, not simply conformism.
- Benchmarks that define nominal operation of the software development process.
- Goals that define the purpose of the metrics program.
- Strategy for achieving the goals.
- An appropriate model (COCOMO, SLIM, etc.) whether it be a mathematical model or heuristic.
- Collection of data that is unobtrusive.
- Analysis of the data to find patterns: patterns imply consistency, and consistency implies process.
- Action on the analysis – change in the process to achieve better results.
- Implementation ethics including trust, value, communication and understanding.

Of course, a record should be kept which documents the efforts of the metrics program and the results.

Measurements are combined into one or more metrics, which help identify the performance of the processes—which are satisfactory and which are not. They are used to measure the progress of a project, the quality of its product, the effort necessary to complete the project, et cetera. One desired outcome of compiling and using these metrics to improve processes is the improvement of the product's value to cost ratio. If a change in a process yields an increase in production during a specific time frame, or yields the same production in a decreased time frame, the value to cost ratio is improved.

Another desired outcome is increased customer satisfaction through meeting their requirements. For example, if defects in software can be traced back to incomplete or faulty requirements definition, the requirements definition process should be reviewed to increase the clarity and completeness of the requirements. The metrics may help show that the customer needs to be more actively involved in defining the requirements clearly and precisely.

The implementation of a metrics program, compiling measurements and statistics, and then using them, is a great method for refining processes and improving the product. Basic measurements, combined into metrics, can help pinpoint faulty processes and help determine a course of action to make them more effective. The result is a product which is delivered in a timely fashion, with few defects, as budgeted.

2.3 Benefits of SCM

There are many benefits to be gained by an organization that practices SCM. Software developers, testers, project managers, Quality Assurance (QA) personnel, and the customer may benefit from SCM. The following is a partial list of some of these benefits.

- Organizes tasks and activities that maintain the integrity of the software.
- Helps manage assets.
- Provides ability to track changes made during sequential or parallel development.

Process Phase	Sample Measurements		
System Design Change Requests	· Title · Description · Program Information · Date and Signature Field · Test Stand Impact · CM Information · Review Board Disposition	· Number · Change Request Cross Reference · Assets Required · Status[2] · Priority[3] · Test Plan · Close-out Action	· Originator · Estimated/Actual Effort[1] · Assigned Engineer(s) · Documentation Readiness · Risk · Peer Review Information
Sub-system Design Change Requests	· *Same as System Design Change Request*		· Type of Change[4]
Software Design Change Requests	· *Same as System/Sub-system Design Change Request* · Source Lines of Code · Memory Impact	· Functional Test Results · Duty Cycle Impact	· List of Modules Changed
Interface Change Requests	· *Same as System Design Change Request* · List of Interface words affected	· Description of each	
Problem Reports (for each Phase)	· *Same as System Design Change Request* · Severity of Error[5] · Reproduction Procedure	· Customer Effect · Development phase when introduced	· Cause of Error
Examples of Derived Metrics	· Man-hours per Project · Schedule Variances · Changes by Type · Tests per Requirement · SLOC per Hour of Effort	· Cycle Time of Project · Change Category Count · Changes by Source · Costs Variances · Return on Investment	· Errors per KSLOC · Requirements Volatility · Average Effort per Change · Defects per Release · Cost Savings
Notes [1]Estimated/Actual Effort example	· Requirements collection · System design · Gathering measurements · Writing software · Debugging software	· Requirements definition · Evaluating errors · Defining metrics · Writing test procedures · Prepare for/attend peer reviews	· Documenting software · Subsystem design · Analyzing metrics · Testing software · Prepare for/perform inspections
[2]Status example	· Proposed · Completed/ Closed-out	· In-analysis · In-rework	· In-implementation · Deferred
[3]Priority example	· High	· Medium	· Low
[4]Type of Change example	· Enhancement · Interface	· Clarification · Error Correction	· New requirement · Error prevention
[5]Severity of Error example	· Show-stopper · Typographical	· Major · Nuisance	· Minor

Table 1. Generic measurements and metrics for a Software Change Process.

- Ensures correct configurations of software, i.e. compatible configurations.
- Ensures that engineers are implementing changes into the correct "baseline" or version of the software.
- Provides the ability to trace the process from requirement to product.
- Limits legal liability by recording all data—whether flattering to the company or not—including memos, decisions, meeting minutes, changes to requirements/code/test procedures, et cetera, providing a "paper trail".
- Helps reduce the lifecycle cost of maintaining or sustaining software (especially for military applications) which can easily exceed the initial cost of development.
- Allows responsibility to be traced to the source; i.e. a requirement problem, coding problem, et cetera.
- Provides for consistent conformance to customer requirements.
- Provides a stable environment for the software development process to be defined, repeated, and improved.
- Enhances compliance with standards being applied.
- Provides an environment in which meaningful measures can be gathered and used.
- Enhances current status accounting.
- Provides data for reports that can be easily generated.
- Allows quick and easy auditing.
- Provides the ability to reproduce circumstances/conditions under which the product was produced by retaining information relative to the production process (tracks changes made to baselines, hardware, compiler versions, etc.).
- Provides communication channels between groups (system, subsystem, test, interface, etc.).
- Fosters an ability to improve without being punitive in nature.
- Provides an understanding of when the product is ready for release (when all changes have been processed completely).
- Helps produce higher quality software.

SCM provides visibility into the status of the evolving software product. Software developers, testers, project managers, Quality Assurance (QA) personnel, and the customer benefit from SCM information.

Software Configuration Management Technology Primer

3

3.1 What is Software Configuration Management?

Software Configuration Management (SCM) is a discipline to manage the evolution of computer program products during all stages of development and sustainment. The Software Engineering Institute Capability Maturity Model (SEI SW-CMM) defines SCM as stated below.

> "[SCM] involves identifying the configuration of the software (i.e., selected software work products and their descriptions) at given points in time, systematically controlling changes to the configuration, and maintaining the integrity and traceability of the configuration throughout the software lifecycle. The work products placed under software configuration management include the software products that are delivered to the customer (e.g., the software requirements document and the code) and the items that are identified with or required to create these software products (e.g., compiler)." [Paulk 93]

SCM provides visibility into the status of the evolving software product. Software developers, testers, project managers, Quality Assurance (QA) personnel, and the customer benefit from SCM information. SCM answers *who, what, when*, and *why*.

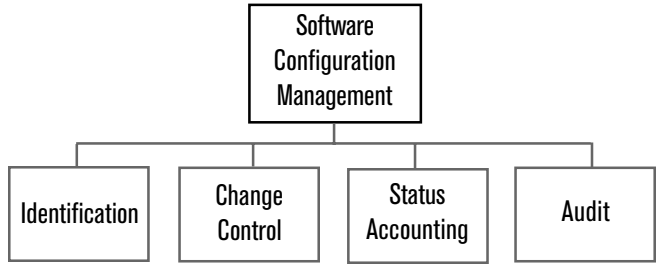
- Who makes the changes?
- What changes were made to the software?
- When were the changes made?
- Why were the changes made?

SCM encompasses the every day tasks within an organization, whether they be software development or sustainment. The software changes are identified, controlled, and managed throughout project lifecycle. Ten key SCM activities for most common development environments are:

- Accessing and retrieving software
- Retrofitting changes across the development lifecycle
- Migrating changes across the development lifecycle
- Managing the compile and build process
- Managing the distribution of changes
- Obtaining approvals and signoffs
- Managing software change requests
- Coordinating communication between groups
- Obtaining project status
- Tracking bugs and fixes [Platinum 98].

SCM is divided into the following functional areas: (1) Identification, (2) Change Control, (3) Status Accounting, and (4) Audit (*see Figure 1*).

Figure 1. Functional Elements of Software Configuration Management.



3.1.1 Configuration Identification

Configuration identification involves identifying the structure of the software system, uniquely identifying individual components, and making them accessible in some form. The goal of configuration identification is to have the ability to identify the components of a system throughout its lifecycle and provide traceability between the software and related software products. Identification answers *What is the configuration of my system? What version of the file is this?* and *What are the components of the system?*

Configuration identification activities include:

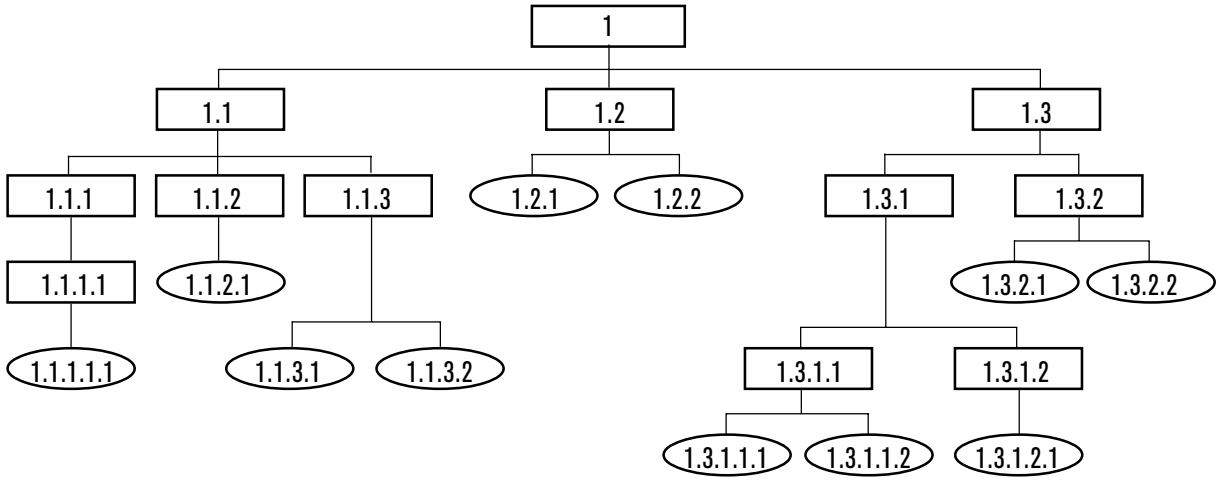
- Selecting items to be placed under SCM control.
- Developing the software hierarchy.
- Creating an identification scheme that reflects the software hierarchy.
- Identifying which version of a component can or cannot be included in a working release.
- Uniquely identifying the various revisions of the software product.
- Defining relationships and interfaces between the various software products.
- Releasing configuration documentation.
- Establishing configuration baselines [MIL-HDBK-61 97].

Figure 2 presents a typical breakdown of software into its distinct parts and presents a numbering scheme that uniquely identifies each component of a baseline release. The number to the left of the dot is the last baseline or major release. The number to the right of the dot is the version since the last baseline or minor release. Normally, after a new baseline, major release, the number to the right of the dot is zero. A hierarchical scheme is used.

Although key components to be managed are the requirements and source code, related documentation and data should be identified and placed under SCM control. It is important to store and track all environment information and support tools used throughout the software lifecycle to ensure that the software can be reproduced. Following are examples of items typically put under SCM control.

- Source code modules.
- System data files.
- System build files/scripts.
- Requirements specifications.
- Interface specifications.
- Design specifications.
- Software architecture specifications.
- Test plans.

Figure 2. Software Configuration Identification Hierarchy.



- Test procedures.
- Test data sets.
- Test results.
- User documentation.
- Software development plan.
- Quality plans.
- Configuration management plans.
- Compilers.
- Linkers and loaders.
- Debuggers.
- Operating systems.
- Shell scripts.
- Third-party tools.
- Other related support tools.
- Procedure language descriptions.
- Development procedures and standards [Kasse 97].

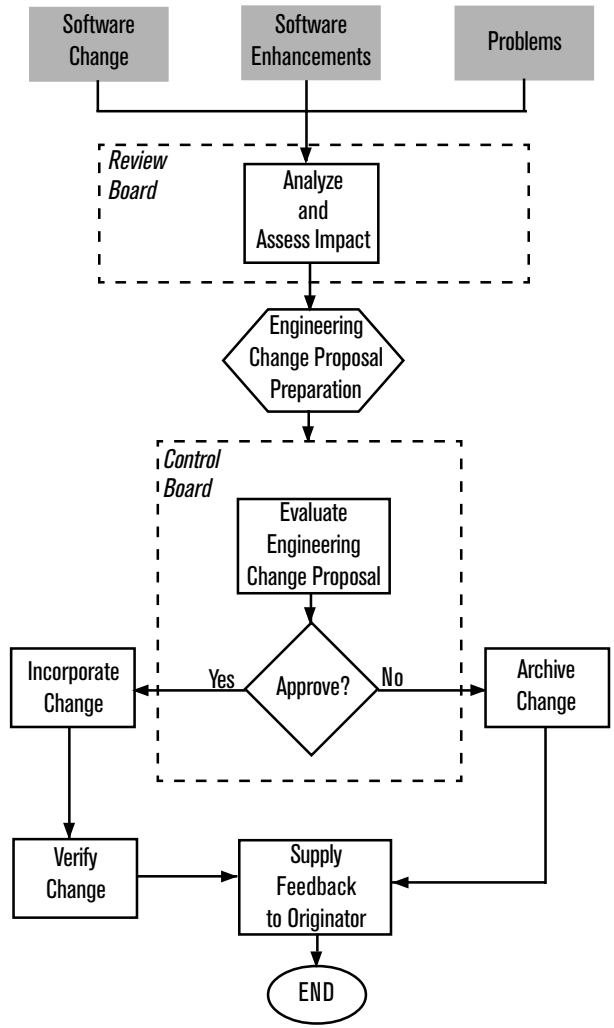
> "Effective configuration identification is a prerequisite for the other configuration management activities (configuration control, status accounting, and audit), which all use the products of configuration identification. If configuration items and their associated configuration documentation are 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 identification of configured items and configuration documentation may result in defective products, schedule delays, and higher maintenance cost after delivery." [MIL-HDBK-61 97]

3.1.2 Configuration Change Control

Configuration change control involves controlling the release and changes to software products throughout the software lifecycle. It "is perhaps the most visible element of configuration management. It is the process to manage preparation, justification, evaluation, coordination, disposition, and implementation of proposed engineering changes and deviations to affected configuration items and baselined configuration documentation." [MIL-HDBK-61 97] The goal of configuration change control is to establish mechanisms that will help ensure the production of quality software. As well as ensure that each version of the software contains all necessary elements, and that all elements in a version will work correctly together. A generic change process is identified in Figure 3.

Configuration change control answers *What is controlled? How are the changes to the products controlled? Who controls*

Figure 3. Generic Change Process [Berlack 92].



the changes? When are the changes accepted, received, and verified?

Configuration change control activities include

- Defining the change process.
- Establishing change control policies and procedures.
- Maintaining baselines.
- Processing changes.
- Developing change report forms.
- Controlling release of the product.

"All changes made to the configuration management baselines or baselined software configuration items should be done according to a documented change control process. The change control process should specify:

- Who can initiate the change requests.
- What the criteria is for placing the software components under formal change control.
- The "change impact" analysis expected for each requested change.
- How revision history should be kept.
- The Check-in/Check-out procedures.
- The process the Software Configuration Control Board (SCCB) follows to approve changes.
- How change requests will be linked to the Trouble Reporting System.
- How change requests are tracked and resolved.
- The reviews and regression tests that must be performed to ensure that changes have not caused unintended effects on the baseline.
- The procedure that will be followed to update all affected software lifecycle components to reflect the approved changes.

"To have effective configuration control that truly supports project development, it is important to establish a change control process that specifies: who can initiate the change requests; the individuals, group, or groups who are responsible for evaluating, accepting, and tracking the change proposals for the various baselined products; the "change impact" analysis expected for each requested change; and how the change history should be kept." [Kasse 97]

In order to control changes made to configuration items or the system, many organizations establish a Software Configuration Control Board (SCCB). The board reviews each proposed change, approves or disapproves it, and if approved, coordinates the change with the affected groups.

Another key concept of change control is the use of baselines. A baseline is "a specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change procedures." [IEEE 90] When an item is baselined, it becomes frozen and can only be changed by creating a new version. Historically three different types of baselines were used: functional, allocated, and product. The functional baseline is the initially approved documentation describing the functional characteristics and the verification required to demonstrate the achievement of those specified functional characteristics. The allocated baseline is the initially approved documentation describing the interface requirements, additional design constraints and the verification required to demonstrate the achievement of those specified functional and interface characteristics. The product baseline is the initially approved documentation describing the necessary functional and physical characteristics and those designated for production acceptance testing. [Berlack 92] In addition, several informal baselines are usually established during the software development process. The number and type of baselines depend on which lifecycle model the project is implementing. Lifecycle models, such as the spiral, incremental development, and rapid prototyping, require more flexibility in the establishment of baselines. For a detailed explanation of lifecycle models, see *Wicked Problems, Righteous Solutions* by Peter DeGrace and Leslie Houlet Stahl. Also see, "A Comparison of Software Development Methodologies," in the January 1995 issue of *CrossTalk: The Journal of Defense Software Engineering.*

3.1.3 Configuration Status Accounting

Configuration status accounting involves the recording and reporting of the change process. The goal of status accounting is to maintain "a continuous record of the status and history of all baselined items and proposed changes to them. It includes reports of the traceability of all changes to the baseline throughout the software lifecycle." [Kasse 97]. Configuration status accounting answers *What changes have been made to the system?* and *How many files were affected by this problem report?*

Configuration status accounting activities include

- Determining type of logs and reports required.
- Tracking the status of SCM items.
- Tracking the status of changes to the system.
- Generating status reports.
- Recording and reporting the activities of SCM.

"Status accounting provides visibility into the system evolution by recording and reporting the status of all items and the status of all requests for change. Questions that SCM status accounting should be able to answer include

- What is the status of an item? A developer may want to know whether a specification has been fully approved. A developer may want to know whether a subsystem has been tested so that the developer can test the modules that interface with that subsystem. A project leader may wish to track the progress of a project as items are developed, reviewed, tested, and integrated.
- Has a change request been approved or rejected by the SCCB?
- Which version of an item implements an approved change request? Once a requested enhancement of a library routine is implemented, the originator and other developers will want to know which version of the routine contains the enhancement.
- What is different about a new version of a system? A new version of a software system should be accompanied by a document listing the changes from the previous version. The change list should include both enhancements and fixes to faults. Any faults that have not been fixed should also be named and described.
- How many faults are detected each month, and how many are fixed? Faults are continuously detected during the operational use of the system. Comparing the number of detected and fixed faults helps to assess the stability of the latest release of the system. Tracking the number of faults also helps the program manager decide when to make a new release of the system.
- What is the cause of the trouble report? Trouble reports can be categorized by their causes: violation of programming standards, inadequate user interface, or invalid, incorrect or incomplete customer requirements. Sometimes when it is discovered that many faults have a similar cause, action can be taken to improve the process and stop such faults from recurring." [Kasse 97]

Key information about the project and configuration items can be communicated to project members through status accounting. Software engineers can see what fixes or files were included in which baseline. Project managers can track completion of problem reports and various other maintenance activities. Minimal reports to be completed include transaction log, change log, and item "delta" report. Other typically common reports include resource usage, "stock status" (status of all configuration items), changes in process, and deviations agreed upon [Ben-Menachem 94].

3.1.4 Configuration Auditing

Configuration auditing verifies that the software product is built according to the requirements, standards, or contractual agreement. Test reports and documentation are used to verify that the software meets the stated requirements. The goal of configuration audit is to verify that all software products have been produced, correctly identified and described, and that all change requests have been resolved according to established SCM processes and procedures. Informal audits are conducted at key phases of the software lifecycle. There are two types of formal audits that are conducted before the software is delivered to the customer: Functional Configuration Audit (FCA) and Physical Configuration

Audit (PCA). FCA verifies that the software satisfies the software requirements stated in the System Requirements Specification and the Interface Requirements Specification. In other words, the FCA allows you to validate the system against the requirements. PCA determines whether the design and reference documents represent the software that was built. Configuration audit answers *Does the system satisfy the requirements?* and *Are all changes incorporated in this version?*

Configuration audit activities include

- Defining audit schedule and procedures.
- Identifying who will perform the audits.
- Performing audits on the established baselines.
- Generating audit reports.

3.1.5 Establishing a Software Baseline Library

In support of the above activities, a software baseline library is established. The library is the heart of the SCM system. It serves as the repository for the work products created during the software lifecycle. Changes to baselines, and the release of software products, are systematically controlled via the change control and configuration auditing functions. The software library

- Supports multiple control levels of SCM.
- Provides for the storage and retrieval of configuration items or units.
- Provides for the sharing and transfer of configuration items or units between control levels within the library.
- Provides for the storage and recovery of archive versions of configuration items or units.
- Helps to ensure correct creation of products from the software baseline library.
- Provides storage, update, and retrieval of SCM records.
- Supports production of SCM reports.
- Provides for maintenance of library structure [Olson 93].

In the past, libraries have been composed of documentation on hard copy and software on machine-readable media. Today, with the advances in information technology and standards that encourage contractors to use automated processing and electronic submittal techniques, organizations are moving toward maintaining all information on machine-readable media.

3.2 Implementing SCM in the Organization

Organizations which have never practiced SCM may find themselves in the unenviable position of needing to begin an SCM program, with no idea where to begin such an effort. The foundation of an SCM program can be set with six building blocks (see Figure 4.):

1. Establish adequate sponsorship
2. Assess current processes
3. Analyze organizational requirements
4. Establish roles and create an SCM team
5. Manage the risks of SCM, and
6. Document the SCM process.

Once these building blocks are in place, results will be seen. These basic principles help to foster the implementation of

Figure 4. Key Steps in Implementing SCM.

a functional and non-instrusive SCM program which will satisfy the needs of management, CM practitioners, design engineers, software engineers, and test engineers, alike.

3.2.1 Establish Adequate Sponsorship

One of the first steps to successfully implement SCM is to obtain management sponsorship. This means public endorsement for SCM and resources needed for success are allocated to the project. Management also needs to establish SCM as a priority and help to facilitate implementation. If management is not willing to "walk the walk," getting the rest of an organization to "walk the walk" will be difficult.

An organization can maintain management sponsorship by identifying and resolving risks, reporting progress, managing SCM implementation details, and communicating with all members of the organization.

3.2.2 Assess Current Processes

The next step is to assess current SCM processes. Every organization that produces software is practicing some sort of SCM. This may not be a formal process or even thought of as SCM. To assess current processes, questions such as the following may be asked: How are files identified? How are versions of software releases identified? How are baselines controlled? What files are included in each release? How are changes to the software identified and tracked? et cetera.

The SCM Key Process Area in the Software Engineering Institute Capability Maturity Model provides structure to evaluate your current processes. It can be very useful in accessing current processes and is a good place to begin.

3.2.3 Analyze Requirements

After assessing your current processes, the next step is to analyze your requirements. What is it that your organization wants to accomplish? The requirement may be a specific level SW-CMM certification, ISO 900 certification, some other standard or certification or simply to improve your software process. Document the requirements for your organization, how you will implement them, and how you will measure success.

3.2.4 Establish Roles and Create an SCM Team

Depending on the requirements for your organization, the various roles and formality of the SCM team may differ. At a minimum there should be a point-of-contact for SCM. The following roles may also be considered.

- Control and review board to analyze and approve changes.
- Software Engineering Process Group (necessary for SW-CMM Level 2 certification) to define processes and practices and facilitate process improvement.
- Project managers and leaders also play a role in SCM in establishing or following SCM plan for their project, ensuring system requirements are properly allocated, ensuring adequate tools are available to support activities and conducting regular reviews.
- A librarian is also necessary to track baselines and versions of files included in each release. An SCM tool may assist in those activities.
- Quality Assurance (QA) may be used to verify documented SCM processes and procedures are followed. QA is also necessary for SW-CMM Level 2 certification.
- Other roles and SCM team members may be identified as required by your organization's SCM.

3.2.5 Manage the Risks of SCM

With each new software project or process, there is some amount of risk associated with it. The same is true when implementing SCM. Whether an organization is implementing a whole new system or just updating a few processes, there will be risks that need to be addressed. Note that having risk is not bad—on the contrary, risk is a necessary part of the SCM and the software development process. Without risk, there is no opportunity for improvement. Risk-free SCM processes are typically of little use.

The very nature of SCM requires risk-taking. Managing and controlling the risks associated with SCM is essential to the success of SCM processes in terms of cost, schedule, and quality. "Risk management costs time and money.

However, it is always less expensive to be aware of and deal with risks than to respond to unexpected problems. A risk that has been analyzed and resolved ahead of time is much easier to deal with than one that surfaces unexpectedly." [Guidelines 96]

The Software Engineering Institute has developed a risk management program comprising six different activities with communication being central to all of them. This program may be used when implementing SCM to effectively manage the associated risks. Risk management should be viewed as an important part of the SCM process. A brief summary of each activity follows.

- **Identify.** Before risks can be managed, they must be identified. Identification surfaces risks before they become problems and adversely affect a project.
- **Analyze.** Analysis is the conversion of risk data into risk decision-making information.
- **Plan.** Planning turns risk information into decisions and actions (both present and future). Planning involves developing actions to address individual risk, prioritizing risk actions, and creating an integrated risk management plan.
- **Track.** Tracking consists of monitoring the status of risks and actions taken to ameliorate risks.
- **Control.** Risk control corrects for deviations from planned risk actions.
- **Communicate.** Risk communication lies at the center of the model to emphasize both its pervasiveness and its criticality. Without effective communication, no risk management approach can be viable. [Paulk 93]

As part of an organization's risk management program, a plan should be developed that integrates the above outlined activities. An SCM risk management plan may focus on addressing risks in three areas: *business, people and technology.* [Burrows 96] The business risks include:

- **Cost.** The expense to incorporate SCM encompasses far more than just the licensing fee for a tool. Management must be willing to make the necessary expenditures for people and resources.
- **Culture shock.** Each organization has its own culture to which the success of the business can be attributed. The procedures and products implemented for SCM must match that culture. The person in charge of SCM needs a broad understanding of software engineering principles and the cultural aspects of the organization.
- **Commitment.** In order to establish a successful SCM process, there must first be a strong commitment from management. The benefits of SCM are not always immediately recognized. "Deploying CM can be a long, costly, and sometimes painful exercise. Counter this risk by building up steam in the project. Get momentum going quickly and keep feeding it." [Burrows 96]

The risks associated with people include:

- **Cheating.** Software developers may try to incorporate their code into the final product without following procedures and resist any changes to the established procedures.
- **Preferred tools.** They may have a tool they want to use that is different from the organization's. To mitigate these risks, try to get offenders to be part of the decision-making process. Let them have input to the procedures and tools that will be used.
- **Resistance.** The greatest barrier to overcome when SCM is introduced into an organization is to change how people view SCM. People generally react negatively toward it. In many organizations, SCM has a low status, and SCM personnel are not trained or qualified to perform their duties. Many software developers perceive SCM as intrusive and have little understanding of the long-term effect of not following SCM procedures. Communication, training, and developer input to SCM processes will help ensure SCM principles are adopted by an organization. [Burrows 96]

The last area is technology. The technology risks include:

- **Loss of control.** At times, it may seem that the SCM procedures and tools are at the controls. There may also be reliance on tools where previously the needed data

and information were obtained manually. Again communication will help to mitigate this risk. Management will have greater control over and information about their projects after successfully implementing SCM.

- **Access.** Controlling who may have access and make changes to various baselines, data repositories, software files or documents is also a risk to be managed. By thorough analysis and design, the procedures implemented may restrict access to approved individuals and give up-to-date information on many aspects of the project that is current and accurate.
- **Scalability.** A project has the potential to outgrow the implemented tool. Counter this risk by selecting a tool that will adapt to the changing size of your organization over time [Burrows 96].

The secret to SCM risk management is to identify and resolve potential risks before they surface unexpectedly or become serious problems. Develop a program for identifying and managing risks. Incorporate an SCM risk management plan that addresses risks to business, people, and technology. Central to everything is communication. Communicate as much as possible to as many people and organizations as possible.

3.2.6 Document the SCM Processes

The SCM requirements identified for your organization will determine the level or degree of documentation needed. Standards or requirements for certification specify what needs to be documented. The assessment performed of your current processes may be included in this step. There are various methods that may be used for documentation from flow charts and diagrams to detailed explanations of each process.

3.3 Writing the Formal SCM Plan

Previous sections of this report describe the elements of SCM or what is included in an SCM process. How those elements are implemented is through a Software Configuration Management Plan. Most organizations follow a policy stating SCM will be done. Detailed procedures outlined in the SCM plan show how the processes mandated by the policy will be carried out. The SW-CMM makes reference to procedures that should be created in order to comply with policies.

The SCM plan specifies what and how SCM shall be done. "While the SCM plan itself is not difficult to write, it is critical to the entire SCM process. The plan provides the focus for the process and procedures, and is the mechanism used to communicate the SCM process to the other organizational groups on the project." [Bounds 96]

The SCM plan identifies

- SCM activities over the software lifecycle.
- SCM organization.
- SCM responsibilities and authority.
- Resources needed to perform SCM functions.
- Interfaces to other organizations.
- SCM roles, policies, and procedures.
- The change control process.
- Level of SCM control.
- Library requirements and activities.
- Members of the Configuration Control Board (CCB).

Reviewing available standards, sample SCM plans, and books will provide the necessary guidance in developing an SCM plan. Standards provide the framework to begin developing an SCM plan. Standards address the key issues that need to be included in the plan. In addition, standards for the development of the SCM process should also be referenced for further guidance in developing an SCM plan [Bounds 96]. For a list of the standards, see Section 3.5.

Sample SCM plans are contained in some of the standards. Various organizations and companies have also published SCM plans that can be used as a model. Books available on SCM plans are included in Appendices A and G.

The most difficult part in preparing an SCM plan is defining the process and writing the procedures. Defining the process entails determining how the elements of SCM will be implemented. For instance, "how will you actually perform change control, what configuration identification scheme will you use, who will need status accounting re-

ports, and what will need to be in each report produced" [Bounds 96]. These are just some of the questions that need to be answered to determine how you will perform SCM on the project.

Once the SCM process is defined, the next step is to write the procedures to invoke this process. One approach to develop your procedures is to document the steps to be executed, then perform the procedure using only the documented steps. This process, documentation and execution, is usually iterated several times to develop a well-defined procedure.

Typically, the procedures are separate from the plan. However, the SCM plan should reference the procedures. "The procedures should describe, step by step, how to do something, whereas the plan should describe what is to be done" [Bounds 96].

The SCM plan is an integral part of a project. The Capability Maturity Model Level 2 Software Configuration Management key process area [see Paulk 95] states that

- The SCM plan is developed in the early stages of and in parallel with the overall project planning.
- The SCM plan is reviewed by the affected groups.

The SCM plan plays a critical role in the success of a project. Therefore, it warrants attention in the early phases of project development. The relative size and complexity of the project do not substantially affect the SCM plan. In a recent study of SCM plans, "no significant differences were noted between a CM plan for a development project versus a maintenance project, a CM plan written for hardware versus software, or a CM plan written for a large project versus a small project.

In general, the same CM plan structure can be used for all of these types of project, with minor adjustments. It was also noted that the majority of differences between the various types of projects exist at the procedure level" [Bounds 96]. Appendix A contains references for sample plans, templates, and checklists used in SCM.

3.4 Automating Software Configuration Management

Automating software configuration management consists of all the steps involved in introducing an SCM tool into an organization and ensuring that it is routinely used on all projects. Implementing an automated SCM system is a complex process. It affects all levels of the organization; therefore, an in-depth evaluation of the organization is required to determine how the processes and people will be affected. Failure to understand the issues involved in the automation of SCM technology is the main reason why organizations do not successfully deploy the SCM tool. A defined strategy that addresses these complex issues becomes a necessity.

Before beginning the automation effort, organizations must consider complex technical issues that may affect the effort. These issues include

- The size and intricacy of the software system.
- Migration to client-server computing.
- Heterogeneous hardware and software platforms.
- Tool integration.
- Legacy systems.
- Interfaces to external systems.

Many organizations thought that purchasing an SCM tool would solve their problems, but soon discovered that there was no "silver bullet" SCM tool. To attempt to automate an immature SCM process will not raise an organization's level of maturity as defined by the SW-CMM. In all likelihood, such attempts would only further amplify any process shortcomings and inadequacies. "Automating a money losing process allows you to lose more money faster and with greater accuracy." [Ventimiglia 97] A tool alone will not solve an organization's SCM problems. Choosing the right tool to satisfy an organization's SCM requirements will in itself fail if other issues are not addressed. To ensure an effective SCM solution, an organization must address the complexities that it faces when implementing a change. "These complexities include

- **Technical.** These issues relate to how the tool operates, how it will be installed to maximize performance and

how it will be customized. For example, how the tool will be installed over the company's network in the client-server architecture given the different platforms and how can it be used to suit the parallel development activities of the various teams.

- **Managerial.** These issues relate to the necessary planning, monitoring, setting of priorities, making of schedules, and resource management. For example, who will be allocated to fulfill the automation activities, how will the product schedules be affected, and who will implement the tool first?
- **Process Related.** These issues relate to the way the company does its business. For example, what is the current flow throughout the company and how do the developers, testers, QA personnel, build managers, document writers, etc., work together to ensure this flow?
- **Organizational.** These issues relate to the infrastructure in the company. For example, how will the tool affect the responsibilities of each department and their intercommunication?
- **Cultural.** These issues relate to the way people operate and achieve their goals. For example, what kind of culture exists at the company, and what is the best way to invoke change in that culture?
- **Political.** These issues relate to "who is stepping on whose toes." For example, how will the organizational boundaries change, who will be responsible for what, and how will people be rewarded based on making the change?
- **People Related.** These issues relate to people's comfort level. For example, how will resistance be managed, and will people lose their job because of this tool? This complexity is closely tied to the cultural.
- **Risk Related.** These issues relate to unknown information and tricky problems. For example, how will the effect of making concurrent changes, such as a new operating system and new hardware, as well as reengineering the legacy code, impact the new SCM system?" [Dart 94]

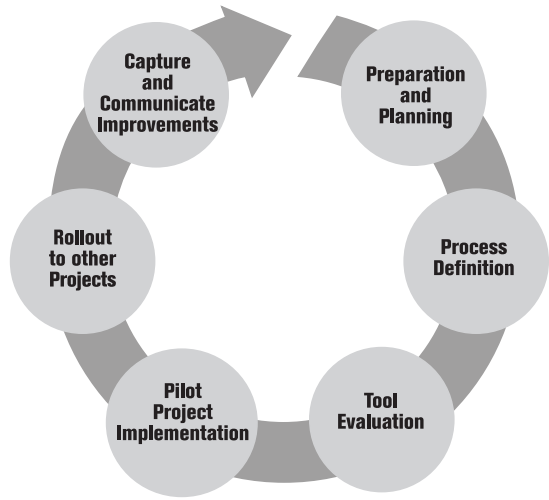
The SCM automation effort must be treated as a project with realistic goals and a defined schedule. SCM automation can be successfully carried out using the phases listed below developed by Susan Dart, a former member of the environment team at the Software Engineering Institute (SEI). The phases provide structured guidance, identify tasks, and address the complexities involved with automating SCM. Key activities are carried out during each phase of the implementation. At all phases, it is important to reinforce management's commitment to the automation effort and to provide training. The phases are as follows (see Figure 4):

Phase 1: Preparation and Planning
Phase 2: Process Definition
Phase 3: Tool Evaluation
Phase 4: Pilot Project Implementation
Phase 5: Rollout to Other Projects
Phase 6: Capture and Communicate Improvements

3.4.1 Phase 1: Preparation and Planning

This is the stage most organizations fail to perform, thereby resulting in the unsuccessful automation of SCM. The purpose of this phase is to plan for the automation activities, to establish management support, and to assess the status of current SCM activities.

Figure 4. Key Phases to Automate SCM.



First, an SCM automation team is created. The automation team is responsible for implementing the automation strategy and plays an important role in the implementation effort. The team monitors and participates in all phases of the automation effort. Members of the automation team typically include

- A leader who is responsible for the automation effort.
- A sponsor who has the authority to empower the team and provide the support required to tackle difficult SCM problems.
- A champion or technical expert who understands the technology.
- Representatives from the user community.

The automation team begins by developing the SCM automation plan. The plan details the benefits of SCM, outlines the automation schedule and resources required, defines the policies and procedures involved in the automation effort, establishes success criteria, and establishes roles of the automation team.

Next, the requirements are defined and prioritized. Developing a clear understanding of the organization's strategic goals is required to evaluate the SCM requirements. The evaluation of SCM requirements should not be conducted in a vacuum. All members of the organization who will be affected by SCM must be surveyed to identify their SCM requirements and to determine their roles in the SCM process. Careful attention must be paid to the training requirements of all people affected by the SCM tool, process, and procedures.

In addition, all levels of management must be aware of the benefits of SCM. Many times this involves showing financial and scheduling benefits, i.e., increase in programmer productivity by automating SCM tasks.

Next, an inventory of present hardware and software platforms is conducted and future hardware and software platforms identified. And, lastly, a risk management plan is developed. This plan identifies risks that could affect the outcome of the automation effort. The automation team is responsible for identifying and addressing risks throughout the project.

3.4.2 Phase 2: Process Definition

The goals of this phase are to define the current SCM process, evaluate the process, and define a new, improved process if required. The process is then analyzed to identify which areas would benefit from automation. A defined software change process is pertinent to the successful implementation of SCM. Without a defined process, the organization will make little progress in the adoption. A variety of methods exists to define the process. Additional information on process definition can be obtained from SEI, IEEE, or the Software Technology Support Center. During this phase, process related requirements will be identified. These should be added to the requirements developed in phase 1 as appropriate.

3.4.3 Phase 3: Tool Evaluation

This phase consists of matching the organization's requirement to SCM tools. Before the evaluation begins, tool requirements identified in phase one are refined and prioritized. The evaluation method is chosen, and test scenarios required to test the capabilities of the tools are developed. It is important to include representatives from all users' groups in the evaluation to gain a better understanding of how different groups will use the tool. Results of a study conducted by the Gartner Group determined that the cost of the software tool represents only 10 percent of the total cost of implementing a solution. Lost productivity accounts for 50 percent and the remaining 40 percent of the solution is derived from the cost of manpower [Softool 92].

Many tool vendors are expanding the functionality of their tools to meet the requirements of today's software development organizations. Several companies sell their products as a series of components. For example, the case product handles version control and process control, whereas the problem reporting function may be purchased separately.

State-of-the-art SCM tools may contain the following features:

- Version control.
- Configuration support.
- Process support.

- Change control.
- Team support.
- Library and repository support.
- Security and protection.
- Reporting and query.
- Tool integration.
- Build support.
- Release management.
- Customization support.
- Graphical user interfaces.
- Distributed development.
- Client-server development support.
- Web support.
- Year 2000 support.

The SCM process should first be defined before tool selection. The tool should implement or automate the defined processes. The tool alone should not be used to define a project's SCM process or procedures.

The tool summary in Appendix B, along with the references, provide a starting point for those responsible for selecting an SCM tool. It may take as long as six months to completely understand the functionality of an SCM tool.

3.4.4 Phase 4: Pilot Project Implementation

The purpose of this phase is to determine how well the SCM tool, processes, and procedures satisfy the organization's requirements. A pilot project allows testing of the tool's functionality on a real project with real data. In addition, the pilot allows for the prototyping of processes and procedures and provides feedback on how users respond to the tool.

It is important to select a pilot that will address all areas of SCM but not affect the project's critical path. The automation team develops standards, policies, and procedures as well as ensures users are trained to perform their SCM duties. Successes and failures are documented and compared to the success criteria identified in the automation plan.

3.4.5 Phase 5: Rollout to Other Projects

This phase involves incremental migration of the tool into other projects. Training and dealing with resistance to change are key activities of this phase. The SCM tool, process, procedures, and training needs are examined and adapted for each project. The automation team implements, evaluates, and monitors rollout activities. This stage is complete when the SCM is routinely used on all projects.

3.4.6 Phase 6: Capture and Communicate Improvements

This phase involves evaluation of automation activities, capturing strategies that worked, and making recommendations for process improvements. The use of measurements and metrics can be very beneficial during this phase.

More details on SCM automation can be found in "Adopting an Automated Configuration Management Solution," by Susan Dart in *Proceedings of the Software Technology Conference,* April 1994.

3.5 SCM Standards

Some of us cringe at the mention of standards; yet, they persist. Why? Because as the software development community strives to become an engineering discipline, the usefulness of standards is recognized.

Standards provide a framework on which acquirer and developer can build a mutual understanding of the acquirer's requirements and of the developer's process.

The table below lists standards and guides that distill years of government and industry experience applicable to organizations that acquire and develop software. Here we have identified some standards that either specifically cover SCM or that cover more general processes that include SCM. Listed are standards specific to SCM as well as hardware configuration management. The standards vary in the breadth and depth of coverage they provide. Scan the "Scope" columns of Table 2 to determine which of the standards listed are specific to SCM, which include general SCM and which merely touch on SCM within the context of broader processes.

Copies of the IEEE/EIA Standards can be purchased from the Institute of Electrical and Electronics Engineers, Inc., IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. Copies may also be purchased from Global Engineering at 800-854-7179. MIL-HDBK-61 is available electronically at http://www.acq.osd.

Military and federal specifications and standards are available free of charge from the Defense Automated Printing Service, Building 4/D, 700 Robbins Avenue, Philadelphia, PA 19111-5094; Voice: 215-697-2179, DSN: 442-2179, Fax: 215-697-1462.

Table 2. Some Standards Related to SCM.

Standards & Guides	Scope			Description	Pages
	Acquisition Process	Supply/Dev Process	Data		
EIA Standard IS-649 National Consensus Std for Configuration Management, Aug. 1995		SCM HCM		Provides basic CM principles and best practices employed by industry to identify product configuration and effect orderly management of hardware and software product change.	60
IEEE Std 1042-1987, Guide to Software Configuration Management (ANSI)		SCM		Describes the application of CM disciplines to the management of software engineering projects. Includes four complete examples of SCM Plans.	90
IEEE Std 828-1990, Standard for Software Configuration Management Plans (ANSI)		SCM		Establishes minimum required contents of an SCM Plan. Supplemented by IEEE Std 1042-1987. Applies to entire life cycle of critical software; also applies to noncritical and already developed software.	15
IEEE/EIA 12207.0-1996, Industry Implementation of International Standard ISO/IEC 12207:1995 (ISO/IEC 12207) Standard for Information Technology - Software Lifecycle Processes, Mar 1998	SCM SW	SCM SW		Establishes a common framework for software lifecycle processes with well-defined terminology. Contains processes, activities, and tasks to be applied during acquisition of a system containing software, stand-alone software product, and software service and during the supply, development, operation, and maintenance of software products. Software includes the software portion of firmware.	75
IEEE/EIA 12207.1-1996, Lifecycle data, April 1998	SCM SW	SCM SW		Provides guidance on what data might be recorded in the execution of the activities and tasks of IEEE/EIA 12207.0-1996. Doesn't dictate content, location, format or media.	30
IEEE/EIA 12207.2-1996, Implementation Considerations, April 1998	SCM SW	SCM SW		Provides guidance in implementing the process requirements of IEEE/EIA 12207.0. Intended to summarize the best practices of the software industry in context of process structure provided by ISO/IEC 12207.	100
ISO 9000-3:1991 (E), Quality Mgmt & Quality Assurance Stds-Part 3: Guidelines for the application of ISO 9001 to the development, supply and maintenance of software		SCM SW		Sets out guidelines to facilitate application of ISO 9001 to organizations that develop, supply and maintain software. Intended to provide guidance where a contract between two parties requires demonstration of a supplier's capability to develop, supply and maintain software products.	15
J-STD-016-1995		SCM SW		Defines a set of software development activities and resulting software products. Provides a framework for software development planning and engineering. As the commercial version of MIL-STD-498, it merges the commercial and Government software development requirements.	220
MIL-HDBK-61, Configuration Management Guidance	SCM SW			Provides guidance and information to DoD acquisition managers, logistics managers, and other individuals assigned responsibility for CM. Assists in planning for and implementing effective DoD CM activities and practices during all life cycle phases of defense systems and configuration items.	200
MIL-STD-2549, Configuration Management Data Interface			Yes	Details the Gov't interface requirements for the exchange of CM information in CM databases. Defines the logical content and relationships of the information that should exist as the information transfers from one activity (and management tool) to another, that is, the business rules view.	500

Key: Software CM = SCM Hardward CM = HCM Software = SW Hardware = HW

Software Configuration Management Case Studies and Lessons Learned

4

4.1 Case Studies

The following case studies outline specific instances where organizations successfully implemented SCM.

4.1.1 Selecting an SCM Tool

In 1988, at a large aerospace corporation in the Southeast, the SCM Manager turned in a recommendation to purchase an SCM automated tool that would satisfy all requirements identified by the SCM groups. Management delayed acting on the recommendation in order to give the other engineering departments time to review the recommended tool.

In the end, the recommendation to purchase the tool was cancelled. It was felt that while the tool did support the SCM organization, it did not adequately address other developmental considerations that the engineering ranks felt were important. Sometime later, a different tool was purchased, one that satisfied all the major requirements of SCM, the software developers, SQA, Test, Integration and management organizations.

4.1.2 Overcoming Barriers to SCM

During a recent visit to a private sector corporation (i.e., they did not deal with government contracts) in New England, it was discovered that the developer's major concern about implementing SCM activities was "all the restrictions" they would have to deal with. They had been led to believe that SCM meant formal controls, restricted access, limited ability to apply creative solutions, and so on. When it was suggested that data can transition to formally controlled baselines through a series of informal control steps, and that SCM did not mean a lock down and bottleneck, they became eager to be involved. After a number of meetings, a phased approach to formal SCM allowed for the placement of informal controls and data gathering which led to baselined items. Everyone was pleased with the process.

The developers soon realized they could work together with SCM as a team to solve problems rather than as two separate organizations with their own concerns and desired solutions. More importantly, perhaps, the SCM group learned that when they got out of their corner office and out onto the engineering floor (being support and service oriented) they quickly became an integral part of the engineering and development process and team.

4.1.3 Implementing SCM with an Electronic Database

A team of 35 to 40 developers was developing six Computer Software Configuration Items (CSCI's) which all had two or more customer variants as well as maintenance variants. The operating system was Unix, and the development languages were Ada, C, and C++. Implementing a classical SCM environment in this type of environment would normally require three to four SCM practitioners to handle all the code and document manipulations. The team chose instead to implement a mostly developer-executed SCM system called *Effective SCM*. They implemented a Software Query Language (SQL) compliant, database driven, pro-

cess-oriented SCM system, which supports a rule-based, closed-loop, change-package approach to development.

Daily interaction with the SCM system by the developers provided 100% tracking and status accounting of everything that happened to any file in the systems without the need for intrusion or interference by SCM practitioners. The SCM practitioners maintained the process model and performed the configured builds. As a result, SCM support to this team was less than one person and in fact is in the order of 80-120 hours per month instead of the over 400 hours per month that a classical approach would have used.

The electronic database created by the engineers completely documented the execution of their software development plan. It also tracks the history of every file used in the system including change documents, baselines and releases for each file. Note: Rule-based, closed-loop change control electronically implements business rules which prevent creation of a new version without authorization, and prevents closure of a change request that hasn't been implemented. A change package approach supports electronic creation of new baselines by application of changes to a previous baseline. The tool electronically adds, replaces, or removes files that are related to the list of changes being made and is very effective in tracking development activities.

(Note that "Effective SCM" is an unregistered trademark of BOBEV Consulting. For a complete description, see "Effective Software Configuration Management" *Crosstalk, The Defense Journal of Software Engineering*, February 1998.)

4.1.4 Obtaining SEI SW-CMM Level 2 Certification

A government organization had been told to reach the SEI SW-CMM Level 2 in 18 months or lose their workload to a contractor. The organization employed an experienced technology transition concern to assist them in reaching Level 2.

Eleven months prior to the SW-CMM assessment the technology transition concern focused on implementing SCM. The approach used was to interview analysts, programmers, and CM specialists and SEPG members to understand the current process. The transition concern met with SEPG members to document the process and to design procedures where procedural holes were identified. A semi-automatic process existed for controlling the software. It was decided that all other aspects of the SCM process would be implemented manually to meet the assessment deadline. If the organization survived the assessment, automation implementation of SCM would proceed.

Because the organization had made previous attempts to establish a more complete SCM process, there was some sceptism about the organization's chances to implement SCM. But the workforce was generally very motivated by the fact that their jobs were at risk.

Like most case studies, this has a happy ending. The organization reached Level 2 and demonstrated significant progress toward Level 3. SCM was the most difficult of the Level 2 Key Process Areas to implement. Obstacles were overcome because: (1) SEPG members played the major role in developing the SCM procedures, rolling out the process, and in auditing the process and the SCM artifacts, and (2) the SEPG was able to effectively leverage off the experience of the technology transition concern.

4.2 Lessons Learned

The following are just a sample of the many lessons that have been learned from apply SCM and its associated technologies. *If any additional information is desired, please contact the authors list at the beginning of this report.*

4.2.1 The Importance of Planning

With only a few exceptions, if you look at any of the SCM standards, manuals, guides, books, etc., you will likely find that SCM has four major functions: (1) identification, (2) change control, (3) status accounting, and (4) audits and verifications. In nearly every case, planning is left out. Yet, SCM is using much more complex equipment to establish and maintain complex environments, multiple baselines, multiple environments on multiple platforms, etc. Like ev-

eryone else, SCM has to do all that faster, cheaper, smarter, and better than before. Planning has become more important than ever.

As recently as 10 years ago, there was still some truth to the statement: "Have them do SCM, someone has to do it and anyone can learn it quickly enough." That is not really true any longer. The job has become too technical, too complex, and dependent on many different variables to make it an, "easy job that anyone can pick up."

It is true that SCM still relies fairly heavily on on-the-job training (OJT). There are no universities or colleges that offer a four-year program in SCM. However, the evolving complexity of the job has been recognized in academia. There are now some two-year community colleges that offer SCM certification programs. Even more surprising, during 1998 one of the authors personally worked with three people who are pursuing doctorate degrees and are centering their respective theses on SCM and its functions.

If this is all true, then "planning" cannot be interpreted as meaning "An SCM Plan has been written." That is certainly a good start, but much more is needed than just a document that explains SCM's roles and responsibilities. SCM planning activities must also include such things as, to name only a few:

- **Metrics.** How long, how many, when and where.
- **Skill Mix.** What is needed and who has it or who can get it.
- **Infrastructure.** Who is doing what, where, when, how.
- **Contingencies.** If this happens, then what.
- **Effort Tracking.** Manpower levels, roll on and roll off.
- **Subcontracts.** Responsibility and authority.
- **Resources.** Budget, tool licenses, training, headcount.
- **Matrix Management.** Decentralized workforce.
- **Control Transitions.** Informal to formal to field.
- **Records Retention.** What gets kept, where, for how long.
- **Control.** Who controls what and how do they do it.
- **Process.** Standardized procedures for repeatability.

4.2.2 Things to Remember

The most significant lessons are:

- Get an inside person on your side—an internal champion. They will become an evangelist for your solution to their co-workers.
- Get management buy-in and sponsorship. Management must really want it, not just go along with it. All levels of management need to support SCM. While implementing SCM, keep a focus on management sponsorship at all times.
- Maintain a sense of humor.
- Be flexible and sensitive to "corporate culture."
- Seek out the early success.
- Don't use critical project as pilot.
- Use a systems approach; for example, where am I, where do I want to go, how am I going to get there?
- Success is more likely with lots of preparation, focus and SCM and developer needs, breadth of participation, on-line access to sample process and planning templates, and standard terminology.
- Keep it simple. If it is too complex, or gets in the way, it will not get used.
- Communicate, communicate, and communicate.

Availability of Templates, Forms, and Checklists

A
Appendix

The following are excellent resources for implementing SCM. Each contains explanations of key SCM concepts. The first three also contain templates, forms and checklists that are commonly used in implementing SCM.

The fourth is an excellent reference for SCM plans and contains a sample plan. The last is an explanation of the SEI SW-CMM.

1. Ben-Menachem, Mordechai, Software Configuration Management Guidebook, McGraw-Hill, Inc., 1994. This book contains sample SCM plans and checklists.

2. Berlack, Ronald H., Software Configuration Management, John Wiley & Sons, New York, 1992. This book contains sample SCM plan outlines, forms, checklists and charts.

3. MIL-HDBK-61, Military Handbook: Configuration Management Guidance, Department of Defense, Sept. 30, 1997. This standard contains sample SCM plans, forms and checklists.

4. Bounds, Nadine M. and Susan A. Dart, *Configuration Management Plans: The Beginning to Your CM Solution,* Software Engineering Institute, Carnegie Mellon University, February 1996.

5. Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrissis, The Capability maturity Model: Guidelines for Improving the Software Process, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., October 1995. This book outlines the SEI SW-CMM and lists and key process areas for each SW-CMM level.

Configuration Management Tool Summary

B
Appendix

The following summarized CM tool information is taken from *Ovum Evaluates: Configuration Management* by Clive Burrows and Ian Wesley published July 1998. It is provided as a courtesy from and with the permission of Ovum Ltd. It is intended to serve as a guide to current and reliable developers of CM tools. The complete 468 page report can be obtained from

Ovum Ltd. 1 Mortimer Street, London W1N 7RH UK
Tel: +44 (0) 171 255 2670 Fax: +44 (0) 171 255 1995

Ovum, Inc. 1 New England Executive Park, Burlington MA 01803
Tel: +1 800 642 OVUM +1 781 272 6414 Fax: +1 781 272 7446

Ovum evaluated the effectiveness of eleven CM tools in the areas of Team Support, Remote Development, Configuration Management, Change Management, Build and Release Support, Process Management, Web Support, Year 2000 Support, Usability, and Administration. This CM Tool Summary identifies the "Key Points," "Strengths," and "When to use" results of that evaluation. It will assist users in building a short list of qualified CM tool providers. It is then the user's responsibility to contact the tool providers of their choice for more detailed product information. For those users desiring it, Ovum's complete 468 page report on these eleven CM tools plus a directory of nineteen other CM tools, would serve as a valuable aid and reference in the selection and deployment of a CM tool.

Name	Company
CCC/Harvest	Platinum Technology
Change Man	Serena Software
ClearCase	Rational Software
Continuus	Continuus Software Corporation
Endevor for MVS	Computer Associates International
PVCS	Intersolv
PCVS Processs Manager	Intersolv
Razor	Tower Concepts, Inc
Source Integrity	Mortice Kern Systems (MKS)
Team Connection	IBM
TRUEchange	TRUE Software

CCC/Harvest

Platinum Technology
1815 South Meyers Road
Oakbrook Terrace IL 60181, USA

Tel: 630-620-5000
Fax: 630-691-0710
http://www.platinum.com

Key points

- Unix and NT clients and servers, plus Windows, Windows 95, and OS/2 clients
- Uses Oracle database for metadata
- Consistent GUI across all platforms

Strengths

- Simple to set up
- Good process control
- Good change package support

When to use

This is a tool best suited for those projects with simple, well-defined, development or maintenance processes. The tool can be readily set up to match existing processes.

Change Man

Serena Software
500 Airport Boulevard, 2nd Floor
Burlingame CA 94010-1904
USA

Tel: 650-696-1800
Fax: 650-696-1849
http://www.serena.com
http://www.optima.com

Key points

- IBM mainframe products with variants tuned to different environments
- Change data held in VSAM control file
- Change package style of working

Strengths

- Good lifecycle support
- Good management of production integrity
- Excellent file merge capability

When to use

A strong contender for many mainframe sites, particularly those with distributed mainframe operations, or those requiring a high degree of integrity for software changes to the production environment. The change package approach and lifecycle processes provide good control over changes. Development and production standards are well supported, but Change Man has the flexibility to support emergency procedures.

Support for maintaining the integrity of production sites as approved change packages are released (including geographically remote sites) is very good.

ClearCase

Rational Software
18880 Homestead Road
Cupertino CA 95014, USA

Tel: 408-863-9900
http://www.rational.com

Key points

- Client-server support on Windows NT and most Unix platforms, with Windows 3.1, 3.11, 95, and NT clients available with attache
- Uses Raima database manager technology
- Transparent to existing development tools and practices

Strengths

- Good support for remote development with MultiSite
- Excellent support for parallel development and file merging
- Good distributed build support, significantly reducing build times

When to use

ClearCase is suited to medium-to-large-scale Windows or Unix development projects, or for organizations migrating from Unix to NT development environments. Smaller teams doing extensive porting work would also benefit. The tool is especially useful for those operating with teams on multiple sites. ClearCase supports heterogeneous environments of PC and Unix systems and, with MultiSite, gives good support for remote development teams.

ClearCase would inflict a heavy burden on small projects, or if applied to small system maintenance activities.

Continuus

Continuus Software Corporation
108 Pacifica
Irvine CA 92618, USA

Tel: 714-453-2200
Fax: 714-453-2276
http://www.continuus.com

Key points

- Repository based on an Informix RDBMS
- Runs on Unix and Windows NT servers, with Unix and Windows NT, 95, and 3.x clients
- Supports parallel development and geographically distributed teams

Strengths

- Good build management support
- Good task process support
- Good web development support

When to use

Mature CM practitioners (those that have progressed through version control) that understand CM and need to implement some measure of process, should highly consider this tool. Aspects that suit Continuus include: (1) enterprise-wide SCM since it can be adapted to fulfill the requirements of most projects; (2) need for strong process support, although some low-level modifications may be necessary to match your own model; and (3) management of Web and intranet sites.

Endevor for MVS

Computer Associates International
One Computer Associates Plaza
Islandia NY 11788
USA
Tel: 800-225-5224, 516-342-5224
Fax: 800-225-5734, 516-342-5329
http://www.cai.com
http://www.cai.com/solutions/year2000/ccm/products.htm
http://www.cai.com/solutions/os390/ccm/

Key points
- IBM mainframe product with variants for client-server and distributed environments
- Integrates with CA's systems management and year 2000 solutions, and other third-party tools
- Part of a large CA product portfolio

Strengths
- Good lifecycle support
- Good management of Production integrity
- Good audit trail of changes

When to use

If you already use Endevor for MVS, then you should continue with it.

CA is concentrating its efforts on integrating its MVS solution with its client-server variants. If this is the direction your organization is heading, then Endevor is a competent solution. Otherwise, you should take a careful look at the alternative companies, which are committed to mainframe CM products.

PVCS

Intersolv
9420 Key West Avenue
Rockville MD 20850, USA
Tel: 301-835-5000
Fax: 301- 838-8064
http://www.intersolv.com

Key points
- Works across heterogeneous LANs
- Runs under Microsoft Windows, OS/2 PM, and Unix
- Has a large and well-established user base

Strengths
- Industry-standard product
- Integrates with a large variety of third-party tools
- Simple and easy to use

When to use

PVCS is well suited to projects where the principal requirement is for Version Management and where parallel development is an occasional need, rather than a regular established way of working. It operates heterogeneously across a wide range of platforms.

PVCS Tracker provides good added capability for users needing integrated change management and problem tracking.

PVCS Process Manager

Intersolv
9420 Keywest Avenue
Rockville MD 20850, USA
Tel: 301-835-5000
Fax: 301-838-8064
http://www.intersolv.com

Key points
- Clients and servers are Windows NT, Unix, VAX and AXP Open VMS, with Windows, Windows 95, and Internet clients
- Uses the Oracle Relational database
- Interoperability with PVCS Version Manager providing transparent process support

Strengths
- Strong control over lifecycle processes, well supported by wizards to aid process definition
- Good change and problem management
- Good web and intranet support

When to use

Process manager is applicable to a wide range of development needs. It is particularly suited to users with a growing need for process automation, and for users with a mixed range of hardware and software environments.

This product is now being sold as part of the PVCS Dimensions suite. Ovum cautions users to check carefully and precisely what features are being offered for sale under the Dimensions name.

Razor

Tower Concepts Inc
248 Main Street
Oneida NY 13421, USA
Tel: 315-363-8000
Fax: 315-363-7488
http://www.tower.com

Key points
- Unix clients and servers plus Windows 95/NT clients
- Own database maintained in RAM and flushed to ASCII files for access by external tools
- Targeted at file version and issue management and CM

Strengths
- Simple to set up
- Good issue management
- Good value for money

When to use

Razor is a tool best suited for projects using a single repository with well-defined development or maintenance processes, and/or where problem tracking and change management are an important requirement. The tool can be readily set up to match the existing processes. If the built-in capability for process support is not sufficient, users can implement their own process using shell scripts and triggers (but see Ovum's reservations about this capability in *Lifecycle support*).

Do not be misled by the low price – this product has a good all-around CM capability.

Source Integrity

Mortice Kern Systems (MKS)
185 Columbia Street West
Waterloo Ontario N2L 5Z5
Canada

Tel: + (1) 519-884-2251
Fax: + (1) 519-884-8861

http://www.mks.com

Key points

- Works across heterogeneous LANs
- Runs under DOS, Microsoft Windows, Windows NT, Windows 95, OS/2, and Unix with an X/Motif GUI
- Has a large and well-established user base

Strengths

- Excellent problem tracking and change management features associated with automatic e-mail notification and reminders between team members
- Good web content management with Web Integrity
- Inexpensive

When to use

Source Integrity is suitable for use on small to medium-sized projects operating over a LAN, where it offers a good all-around capability for most CM needs, provided that parallel development is an occasional requirement rather than an established way of working.

Ovum recommends that users consider Source Integrity Professional Edition to gain the full associated benefits of problem tracking and change management support.

Webmasters should give serious consideration to the management support provided by Web Integrity.

TeamConnection

IBM Direct Sales
Attn. Linda Davis
7100 Highlands Parkway
Smyrna GA 30082

Tel: 800-426-2255 Ext. 31825

http://www.software.ibm.com/ad/teamcon
Or contact any IBM office worldwide

Key points

- AIX, HP-UX, Solaris, OS/2 & Windows NT clients and servers, plus Windows 95 clients
- Uses IBM's DB2 Universal Database
- Supports electronic deployment of application software

Strengths

- Excellent merge tool for parallel development
- Good defect and change management
- Good build support

When to use

A good CM tool with good all-around capability for most development team requirements, but not suited to remote development with closed repositories.

TRUEchange

TRUE Software
300 Fifth Avenue
Waltham MA 02451

Tel: 781-890-4450
Fax: 781-890-4452
http://www.truesoft.com

Key points

- Unix, NT and VMS servers; with Windows (all variants), Unix, NT, VMS, and MVS clients
- Uses its own internal repository system
- Originator of change-set methodology

Strengths

- Good management level reporting
- Good release management and conflict detection
- Good change management

When to use

TRUEchange is ideally suited for managing the on-going flow of changes to production applications, particularly in large IT organizations moving mission-critical systems to the distributed world.

It is not limited to maintenance projects. The principles are always applicable, but the product would have fewer competitors in a maintenance context than in a development context. TRUEchange is available on a wide range of platforms and will be of special interest to users with heterogeneous development environments.

TRUEchange with TRUEtrack has a good all-around SCM capability, and should not be viewed solely in the context of the change set technology.

TRUErelease extends the product range capability to establish the integrity of applications before they are distributed for production use.

Training and Education

C
Appendix

Software Technology Support Center (STSC)

OO-ALC/TISE
7278 Fourth St.
Hill AFB, UT 84056-5205
Voice: Paul Hewitt, DSN 775-5742 (801)775-5742
Reed Sorensen, DSN 775-5738 (801)775-5738
Russ Lee, DSN 775-5740 (801)775-5740
Fax: DSN 777-8069 (801)777-8069
Internet: hewittp@software.hill.af.mil
sorenser@software.hill.af.mil
leeru@software.hill.af.mil

STSC offers the following services:

- Configuration Management Workshop
- Capability Evaluation
- Executive Session
- Tools Evaluation, and
- Other consulting services that assist software development, maintenance, and acquisition organizations

Software Engineering Institute (SEI)

Carnegie Mellon University
Pittsburgh, PA 15213-3890
Voice: 412-268-5800
E-mail: customer-relations@sei.cmu.edu
Internet: http://www.sei.cmu.edu

SEI offers the following courses:

- CBA Lead Assessor Training
- Defining Software Processes
- Implementing Goal-Driven Software Measurement
- Introducing New Software Technology
- Introduction to the Capability Maturity Model for Software
- Introduction to the People Capability Maturity Model
- Introduction to the Software Acquisition Capability Maturity Model

IEEE

445 Hoes Lane
P.O. Box 1331
Piscataway, N.J. 08855-1331
Voice:732-562-3811
Fax: 732-562-1571
E-mail: p.gerdon@ieee.org

IEEE offers the following courses (some on-site):

- Software Configuration Management
- Software Project Management
- Software Quality Assurance
- Software Requirements Specifications
- Software Reviews and Audits
- Software Testing
- Software Verification & Validation

Abelia Corporation

12224 Grassy Hill Court
Fairfax, VA 22033-2819
Voice: 703-591-5247
Internet: http://www.abelia.com/sep98.htm

- Configuration Management

Configuration Management Training Foundation

Voice: 530-873-2734
Fax: 530-873-4835
E-mail: tracie@cmtf.com

The following courses are offered:

- Configuration Management
- Advanced Configuration Management
- Software Configuration Management
- Intergrated Product Teams

InRoads Technology, Inc

130 Robin Hill Rd. Suite #120
Santa Barbara, CA 93117
E-mail: info@inroadstech.com
Voice: (805) 967-4545
Fax: (805) 964-4790

- Fundamentals of Software Configuration Management

Institute of Configuration Management

P.O. Box 5656
Scottsdale, AZ 85261-5656
E-mail: info@icmhq.com
Voice: (602) 998-8600
Fax: (602) 998-8923

Courses offered:

- CMII-Based Business Process Infrastructure
- Structured Configuration and Process Information
- Change Forms, Effectivities and Traceability
- Change Administration and Change Processing
- Enterprise-Wide CM Plan and Procedures
- Process Improvement Plan and Implementation
- Software Configuration Management
- Software CM Process Improvement

Integrated Support Systems

Voice: 864-654-1284, extension 110
E-mail: training@isscorp.com

- Configuration Management

Learning Tree

Voice: 800-843-8733

- Software Configuration Management

Lexington Software Associates

6 New England Executive Park
Suite 400
Burlington, MA 01803
Voice: 978-266-2730
Fax: 978-266-2984
E-mail: sales@lsai.com

- Courses focusing on ClearCase for CM

Patriate Limited

Eagle House
The Ring
Bracknell
Berkshire. RG12 1HB
Voice: 01-344 382111
Fax: 01-344 304728
E-mail: info@patriate.com

- Courses focusing on ClearCase, ClearDDTS, and ClearGuide

System Technology Institute, Inc.

P.O. Box 6907
Malibu, CA 90264-6907
E-mail: STIclass@aol.com
Voice: 310-457-0851
Fax: 310-457-0951

- Courses on Software Engineering, Management, and Assurance Training

Technology Training Corporation

3420 Kashiwa Street
Los Angeles, California 90505
E-mail: ttchq@ttcus.com
Voice: (310) 534-3922
Fax: (310) 534-2964

- Contact organization for specific course information

Value Added Systems Technology, Inc

Voice: 770-988-2792
info@vastcorp.com

- Contact organization for specific course information

Software Technology Support Center's SCM Services

D
Appendix

STSC Introduction

In 1987, the U.S. Air Force selected Ogden Air Logistics Center at Hill Air force Base, Utah to establish and operate its Software Technology Support Center (STSC). It was chartered to be the command focus for pro-active application of software technology in weapons, command and control, intelligence, and mission-critical systems.

The STSC provides hands-on assistance in adopting effective technologies for software-intensive systems. We help organizations identify, evaluate, and adopt technologies that improve software product quality, production efficiency, and predictability.

We use the term *technology* in its broadest sense to include processes, methods, techniques, and tools that enhance human capability. Our focus is on field-proven technologies that will benefit the DoD mission.

SCM Services

The STSC Configuration Management (CM) team provides consultation services for U.S. Air Force and other governmant organizations to understand, evaluate, and adopt software configuration management technologies with the goal of improve productivity, predictability, and maturity of their software processes.

The CM team is committed to helping you reach and maintain your software process improvement objectives. We will help you formulate a systematic strategy for success through on-site evaluations and discussions with key personnel, make recommendations for changes and improvements, and most important, we will be there to guide your organization step by step through the entire software process improvement effort.

The CM Team comprises government civilian and contractor employees located at Hill Air Force Base. Contractor support is provided by TRW. Experience averages 20 years for each team member in software development, documentation, acquisition, planning, configuration management, operating systems, and programming in Assembly, COBOL, FORTRAN, JOVIAL J3B, and J73, Ada, C, and C++. Team members have applied the processes and methods covered by DOD-STD-2167A, MIL-STD-498, J-STD-016-1995, MIL-STD-1553, and MIL-STD-1553B.

Successful adoption of CM is a complex task. The STSC's approach is to team with software organizations. Periodic review by the STSC every four to six weeks focuses SEPG members, management, and practitioners on the adoption effort. The support and guidance that is provided by an STSC representative is a key to maintaining focus and achieving success.

The STSC provides the following SCM services:

- **Configuration Management Workshop.** The CM Workshop is designed for software development and maintenance organizations that are interested in succeeding at CM, that want to achieve process maturity as specified in the SEI SW-CMM, and that need to understand CM as specified in both commercial and military standards.
- **Capability Evaluation.** This service is designed for software development and maintenance organizations that

are interested in implementing CM or want to achieve CM process maturity.

- **Executive Session.** The CM Executive Session provides basic education and orientation to senior and upper midlevel managers about CM. Each session is customized to the needs of the executives. They can select which topics are of interest to them.
- **Tools Evaluation.** Once an organization has developed an effective process and determined its methods, the selection and use of software tools is appropriate. The STSC tool evaluation method addresses the typical issues of incomparable evaluations, marketing hype, and individual biases.

Overview of SEI's Software Capability Maturity Model

E
Appendix

The following material is taken from SEI SW-CMM: Guidelines for Improving the Software Process. [Paulk 95]

Overview

"The Capability Maturity Model for Software developed by the SEI is a framework that describes the key elements of an effective software process. The SW-CMM describes an evolutionary improvement path for software organizations from an ad hoc, immature process to a mature, disciplined one. This path is encompassed by five levels of maturity.

"The SW-CMM guides software organizations that want to gain control of their processes for developing and maintaining software and to evolve toward a culture of software engineering and management excellence. Its purpose is to guide these organizations in selecting process improvement strategies by determining their current process maturity and identifying the few issues most critical to improving their software process and software quality.." [Paulk 95] The following table lists the key process areas and characteristics for each level.

Table 1. SW-CMM Levels and Characteristics [Paulk 95]

Level	Characteristic	Key Process Area	
1 Initial	The software process is characterized as adhoc and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.		
2 Repeatable	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on project with similar applications.	Requirements management Software project planning Software project tracking & oversight	Software subcontract management Software quality assurance Software configuration management
3 Defined	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for development and maintaining software.	Organization process focus Organization process definition Training program Integrated software management	Software product engineering Intergroup coordination Peer reviews
4 Managed	Detailed measure of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.	Quantitative process	Software quality management
5 Optimizing	Continuous process improvement is enabled by quantitiatve feedback from the process and for piloting innovative ideas and technologies.	Defect prevention	Technology change management

SCM in the CMM

The SCM KPA consists of the following goals, commitment, abilities, activities, measurement and verifications. They are specified in [Paulk 95]. The goals are:

1. SCM activities are planned
2. Selected software work products are identified, controlled, and available
3. Changes to identified software work products are controlled, and
4. Affected groups and individuals are informed of the status and content of software baselines.

The commitment is the project follows a written organizational policy for implementing SCM.

The abilities consist of the following:

1. A board having the authority for managing the project's software baselines exists or is established
2. A group that is responsible for coordinating and implementing SCM for the project exists
3. Adequate resources and funding are provided for performing the SCM activities, and
4. Members of the SCM group are trained in the objectives, procedures, and methods for performing their SCM activities.

The activities are:

1. An SCM plan is prepared for each software project according to a documented procedure
2. A documented and approved SCM plan is used as the basis for performing the SCM activities
3. A configuration management library system is established as a repository for the software baselines
4. The software work products to be placed under configuration management are identified
5. Change requests and problem reports for all configuration items/units are initiated, recorded, reviewed, approved, and tracked according to a documented procedure
6. Changes to baselines are controlled according to a documented procedure
7. Products from the software baseline library are created and their release is controlled according to a documented procedure
8. The status of configuration items/units is recorded according to a documented procedure
9. Standard reports documenting the SCM activities and the contents of the software baseline are developed and made available to affected groups and individuals, and
10. Software baseline audits are conducted according to a documented procedure.

The measurement is measurements are made and used to determine the status of the SCM activities.

The verifications are:

1. The SCM activities are reviewed with senior management on a periodic basis
2. The SCM activities are reviewed with the project manager on both a periodic and event-driven basis
3. The SCM group periodically audits software baselines to verify that they conform to the documentation that defines them, and
4. The software quality assurance group reviews and/or audits the activities and work products for SCM and reports the results.

Acronyms and Glossary

F
Appendix

Acronyms

AF	Air Force
AFSCM	Air Force Systems Command Manual
ASCII	American Standard Code for Information Interchange
ASSET	Asset Source for Software Engineering Technology
CALS	Computer-Aided Acquisition and Logistics Support
CASE	Computer-Aided Software (or Systems) Engineering
CI	Configuration Item
CCB	Configuration Control Board
CDR	Critical Design Review
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CM	Configuration Management
CMM	Capability Maturity Model
CMU	Carnegie Mellon University
DARPA	Defense Advanced Research Projects Agency
DoD	Department of Defense
ECS	Electronic Customer Services
EIA	Electronic Industry Association
ESIP	Embedded Computer Resources Support Improvement Program
FCA	Functional Configuration Audit
GUI	Graphical User Interface
HCM	Hardware Configuration Management
IEEE	Institute of Electrical and Electronics Engineers
IPSE	Integrated Project Support Environment
ISO	International Organization for Standardization
KPA	Key Process Area
LAN	Local Area Network
MIS	Management Information System
NASA	National Aeronautics and Space Administration
PC	Personal Computer
PCA	Physical Configuration Audit
PCTE	Portable Common Tool Environment
PDR	Preliminary Design Review
QA	Quality Assurance
RCS	Revision Control System
SCCB	Software Configuration Control Board
SCCS	Source Code Control System
SCML	Software Configuration Manager Library (IBM)
SCM	Software Configuration Management
SDP	Software Development Plan
SDR	System Design Review
SEI	Software Engineering Institute
SEPG	Software Engineering Process Group
SQA	Software Quality Assurance
SQL	Software Query Language
S/SEE	System/Software Engineering Environment
STARS	Software Technology for Adaptable Reliable Systems
STC	Software Technology Conference
STD	Standard
STSC	Software Technology Support Center
SW-CMM	Capability Maturity Model for Software
VDD	Version Description Document
WWISCUC	World-Wide Information System Common User Contract
WWMCCS	World-Wide Military Command and Control System

Glossary

Unless otherwise noted, all references are from [IEEE 90].

Allocated Baseline. The initial approved specifications governing the development of configuration items that are part of a higher level configuration item.

Architecture. The organizational structure of a system or component.

Audit. An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria.

Baseline. A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

Capability Maturity Model (CMM). A description of the stages through which software organizations evolve as they define, implement, measure, control, and improve their software processes. This model provides a guide for selecting process improvement strategies by facilitating the determination of current process capabilities and the identification of the issues most critical to software quality and process improvement [Paulk 93].

Checkout . Testing conducted in the operational or support environment to ensure that a software product performs as required after installation.

Complexity. The degree to which a system or component has a design or implementation that is difficult to understand and verify.

Component. One of the parts that make up a system. A component may be hardware or software and may be subdivided into other components.

Computer-Aided Software Engineering (CASE). The use of computers to aid in the software engineering process. May include the application of software tools to software design, requirements tracing, code production, testing, document generation, and other software engineering activities.

Computer Software Component (CSC). A functionally or logically distinct part of a computer software configuration item, typically an aggregate of two or more software units.

Computer Software Configuration Item (CSCI). An aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

Configuration. The functional and physical characteristics of hardware or software as set forth in technical documentation or achieved in a product.

Configuration Control. An element of configuration management that consists of the evaluation, coordination, approval or disapproval, and implementation of changes to configuration items after formal establishment of their configuration identification.

Configuration Control Board (CCB). A group of people who evaluates and approves or disapproves proposed changes to configuration items and ensures implementation of approved changes.

Configuration Identification. An element of configuration management that consists of selecting the configuration items for a system and recording their functional and physical characteristics in technical documentation.

Configuration Item (CI). An aggregation of hardware or software or both that is designated for configuration management and treated as a single entity in the configuration management process.

Configuration Item Development Record. A document used in configuration management that describes the development status of a configuration item based on the results of configuration audits and design reviews.

Configuration Management (CM). A discipline that applies technical and administrative direction and surveillance to identify and document the functional and physical characteristics of a configuration item, control changes to those characteristics, record and report change processing and implementation status, and verify compliance with specified requirements.

Configuration Status Accounting. An element of configuration management that records and reports information needed to manage a configuration effectively. This information includes a listing of the approved configuration identification, the status of proposed changes to the configuration, and the implementation status of approved changes.

Critical Design Review (CDR). A review conducted to verify that the detailed design of one or more configuration items satisfies specified requirements; to establish the compatibility among the configuration items and other items of equipment, facilities, software, and personnel; to assess risk areas for each configuration item; and as applicable, to assess the results of producibility analyses, review preliminary hardware product specifications, evaluate preliminary test planning, and evaluate the adequacy of preliminary operation and support documents.

Functional Baseline. The initial approved technical documentation for a configuration item.

Functional Configuration Audit (FCA). An audit conducted to verify that the development of a configuration item has been satisfactorily completed, that the item has achieved the performance and functional characteristics specified in the functional or allocated configuration identification, and that its operational and support documents are complete and satisfactory.

Hierarchy. A structure in which components are ranked into levels of subordination; each component has zero or one or more subordinates, and no component has more than one superordinate component.

Interface. A hardware or software component that connects two or more other components to pass information from one to the other.

Key Process Area (KPA). A cluster of related activities that, when performed collectively, achieve a set of goals that are important for establishing process capability. The key process areas have been defined to reside at a single maturity level. They are the areas identified by the SEI to be the principal building blocks to help determine the software process capability of an organization and understand the improvements needed to advance to higher maturity levels [Paulk 93].

Maturity Level. A well-defined evolutionary plateau toward achieving a mature software process. The five maturity levels in the SEI's Capability Maturity Model are *initial, repeatable, defined, managed, and optimizing* [Paulk 93].

Module. A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading; for example, the input to or the output from an assembler, compiler, linkage editor, or executive routine.

Physical Configuration Audit (PCA). An audit conducted to verify that a configuration item as built conforms to the technical documentation that defines it.

Preliminary Design Review (PDR). A review conducted to evaluate the progress, technical adequacy, and risk resolution of the selected design approach for one or more configuration items; to determine each design's compatibility with the requirements for the configuration item; to evaluate the degree of definition and assess the technical risk associated with the selected manufacturing methods and processes; to establish the existence and compatibility of the physical and functional interfaces among the configuration items and other items of equipment, facilities, software, and personnel and, as applicable, to evaluate the preliminary operational and support documents.

Product Baseline. The initial approved technical documentation (including, for software, the source code listing) defining a configuration item during the production, operation, maintenance, and logistic support of its lifecycle.

Product Configuration Identification. The current approved or conditionally approved technical documentation that defines a configuration item during the production, operation, maintenance, and logistic support phases of its lifecycle. It prescribes all necessary physical or form, fit, and function characteristics of a configuration item, the selected functional characteristics designated for production acceptance testing, and the production acceptance tests.

Project Plan. A document that describes the technical and management approach to follow for a project. The plan typically describes the work to be done, the resources required, the methods to use, the procedures to follow, the schedules to meet, and the way that the project will be organized.

Quality Assurance (QA). A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements.

Release. The formal notification and distribution of an approved version.

Software Configuration Control Board (SCCB). *See Configuration Control Board.*

Software Development Plan (SDP). A collection of plans that describe the activities to be performed for the software project. It governs the management of the activities performed by the software engineering group for a software project [Paulk 93].

Software Engineering. The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software.

Software Engineering Environment. The hardware, software, and firmware used to perform a software engineering effort. Typical elements include computer equipment, compilers, assemblers, operating systems, debuggers, simulators, emulators, test tools, documentation tools, and database management systems.

Software Engineering Group. A group comprising both managers and technical staff who have responsibility for software development and maintenance activities, i.e., requirements analysis, design, code, and test, for a project. Groups performing software-related work, such as the Software Quality Assurance Group, the Software Configuration Management Group, and the Software Engineering Process Group, are not included in the Software Engineering Group [Paulk 93].

Software Engineering Process Group (SEPG). A group of specialists who facilitate the definition, maintenance, and improvement of the software process used by the organization [Paulk 93].

Software Library. A controlled collection of software and related documentation designed to aid in software development, use, or maintenance. Types include master library, production library, software development library, software repository, and system library.

Software Lifecycle. The period that begins when a software product is conceived and ends when the software is no longer available for use. The software lifecycle typically include a concept phase, requirements phase, design phase, implementation phase, test phase, installation and checkout phase, operation and maintenance phase, and sometimes, retirement phase.

Software Process Maturity. The extent to which a specific process is explicitly defined, managed, measured, controlled, and effective. Maturity implies a potential for growth in capability and indicates both the richness of an organization's software process and the consistency with which it is applied in projects throughout the organization [Paulk 93].

System Design Review (SDR). A review conducted to evaluate the manner in which the requirements for a system have been allocated to configuration items, the system engineering pro-

cess that produced the allocation, the engineering planning for the next phase of the effort, manufacturing considerations, and the planning for production engineering.

Version. An initial release or re-release of a computer software configuration item, associated with a complete compilation or recompilation of the computer software configuration item. Also for documentation, an initial release or complete re-release of a document, as opposed to a revision resulting from issuing change pages to a previous release.

Version Description Document (VDD). A document that accompanies and identifies a given version of a system or component. Typical contents include an inventory of system or component parts, identification of changes incorporated into this version, and installation and operating information unique to the version described.

Bibliography

G
Appendix

[**ANSI/IEEE 87**] ANSI/IEEE Std 1042-1987, *American National Standard IEEE, Guide to Software Configuration Management,* Institute of Electrical and Electronics Engineers, Inc., New York, N.Y., 1988.

[**Ayer 92**] Ayer, Steve J., and Frank S. Patrinostro, *Software Configuration Management: Identification, Accounting, Control, and Management,* McGraw-Hill Software Engineering Series, McGraw-Hill, 1992.

[**Babich 86**] Babich, Wayne A., *Software Configuration Management: Coordination for Team Productivity,* Addison-Wesley, 1986.

[**Ben-Menachem 94**] Ben-Menachem, Mordechai, *Software Configuration Management Guidebook*, McGraw-Hill, 1994.

[**Berlack 92**] Berlack, Ronald H., *Software Configuration Management*, John Wiley & Sons, New York, 1992.

[**Boehm 81**] Boehm, Barry, "Software Engineering Economics", 1981.

[**Bounds 96**] Bounds, Nadine M. and Susan A. Dart, *Configuration Management Plans: The Beginning to Your CM Solution,* Software Engineering Institute, Carnegie Mellon University, February 1996.

[**Bray 95**] Bray, Olin and Michael M. Hess, "Reengineering a Configuration Management System," IEEE Software, January 1995.

[**Buckley 92**] Buckley, Fletcher J., *Implementing Configuration Management: Hardware, Software, and Firmware*, IEEE Computer Society Press, Los Alamitos, Calif., 1993.

[**Buckley 94**] Buckley, Fletcher J., "Implementing a Software Configuration Management Environment," *IEEE Computer*, 1994.

[**Butler 95**] Butler, Kelley L., "The Economic Benefits of Software Process Improvement", *CrossTalk, The Defense Journal of Software Engineering,* Software Technology Support Center, July 1995.

[**Burrows 96**] Burrows, Clive, George W. George, and Susan Dart, *Ovum Evaluates Configuration Management,* Ovum Limited, 1996.

[**Burrows 98**] Burrows, Clive, and Ian Wesley, *Ovum Evaluates Configuration Management,* Ovum Limited, 1998.

[**Carnegie 98**] Carnegie Mellon University and the Software Engineering Institute, "The '98 Software Engineering Symposium Preliminary Program", 1998.

[**Carr 93**] Carr, M., S. Kondra, I. Monarch, F. Ulrich, and C. Walker, *Taxonomy-Based Risk Identification*, Technical Re-

port CMU/SEI-93-TR-6, Software Engineering Institute, Carnegie Mellon University, 1996.

[**Conner 82**] Conner, Daryl R. and Robert W. Patterson, "Building Commitment to Organization Change," *Training and Development Journal*, Vol. 36, No. 4, April 1982, pp. 18-30.

[**Dart 90a**] Dart, Susan A., "Issues in Configuration Management Adoption," *Proceedings of Conference on Caseware,* Software Engineering Institute Overview, Carnegie Mellon University, Pittsburgh, Pa., 1990.

[**Dart 90b**] Dart, Susan A., *Spectrum of Functionality in Configuration Management Systems,* Technical Report CMU/SEI-90-TR-11, ESD-90-TR-212, Software Engineering Institute, Carnegie Mellon University, 1990.

[**Dart 92a**] Dart, Susan A., "State-of-the-Art in Environment Support for Configuration Management," *ICSE 14 Tutorial*, Australia, Carnegie Mellon University, Pittsburgh, Pa, May 1992.

[**Dart 92b**] Dart, Susan A., *The Past, Present, and Future of Configuration Management*, Technical Report CMU/SEI-92-TR-8, ESC-TR-92-8, Software Engineering Institute, Carnegie Mellon University, July 1992.

[**Dart 94**] Dart, Susan A., "Adopting an Automated Configuration Management Solution", *Proceedings of Software Technology Conference*, April 1994.

[**Dart 96**] Dart, Susan, A., "Achieving the Best Possible Configuration Management Solution," *CrossTalk, The Defense Journal of Software Engineering,* Software Technology Support Center, Hill Air Force Base, UT, September 1996.

[**DeGrace 90**] DeGrace, Peter and Leslie Hulet Stahl, "Wicked Problems, Righteous Solutions," *A Catalogue of Modern Software Engineering Paradigms*, Yourdon Press, Englewood Cliffs, N.J., 1990.

[**Evans 97**] Evans, Michael W. and Shawn T. O'Rourke, "CenterZone Management: The Relationship Between Risk Management and Configuration Management in a Software Project", *Proceedings of Software Technology Conference*, April 1997.

[**Feiler 91**] Feiler, Peter H., *Configuration Management Models in Commercial Environments,* Technical Report CMU/SEI-91-TR-7, ESD-9-TR-7, Software Engineering Institute, Carnegie Mellon University, March 1991.

[**Feiler 90**] Feiler, Peter H. and Grace Downey, *Transaction-Oriented Configuration Management: A Case Study*, Technical Report CMU/SEI-90-TR-23, ESD-90-TR-224, Software Engineering Institute, Carnegie Mellon University, November 1990.

[**Firth 87**] Firth, Robert, et al., *A Guide to the Classification and Assessment of Software Engineering Tools*, Technical Report CMU/SIE-87-TR-10, ESD-TR-87-111, Software Engineering Institute, Carnegie Mellon University, August 1987.

[**Forte 90**] Forte, Gene, "Configuration Management Survey," *CASE Outlook* 90(2), 1990.

[**Fowler 88**] Fowler, Pricilla and Stan Przybylinski, "Transferring Software Engineering Tool Technology," IEEE Computer Society Press, Washington, D.C., 1988.

[**Guidelines 96**] *Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems*, Software Technology Support Center, Hill Air Force Base, UT, June 1996.

[**Haque 97**] Haque, Tani, "Process-Based Configuration Management: The Way to Go to Avoid Costly Product Recalls," *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, April 1997.

[**Hermann 98**] Hermann, Brian and Russell, Jim, Marshall, "Are You Ready to Deliver? To Ship? To Test?" *CrossTalk, The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, August 1998.

[**Humphrey 90**] Humphrey, Watts S., *Managing the Software Process*, Addison-Wesley, August 1990.

[**ICM 98**] Institute of Configuration Management, CMII model, Course I "CMII-Based Business Process Infrastructure".

[**IEEE 90**] IEEE Std 828-1990, *IEEE Standard for Software Configuration Management Plans,* 1990.

[**Kasse 97**] Kasse, Tim, "Software Configuration Management for Project Leaders", *Proceedings of Software Technology Conference*, April 1997

[**Kingsbury 96**] Kingsbury, Julie, "Adopting SCM Technology," *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, March 1996.

[**Marshal 95**] Marshall, A.J., "Demystifying Software Configuration Management," *CrossTalk*, The Defense Journal of Software Engineering, Software Technology Support Center, Hill Air Force Base, UT, May 1995.

[**Marshal 95**] Marshall, Alexa J., "Software Configuration Management: Function or Discipline?," *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, October 1995.

[**MIL-HDBK-61 97**] MIL-HDBK-61, *Military Handbook: Configuration Management Guidance,* Department of Defense, Sept. 30, 1997.

[**Mosley 95**] Mosley, Vicky, "Improving Your Process for the Evaluation and Selection of Tools and Environments," *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, September 1995.

[**Myers 95**] Myers, Robin J., "Configuration Management: A Prerequisite for BPR Success," Enterprise Reengineering, August 1995.

[**Olson93**] Olson, Timothy G., et al., *A Software Process Framework for the SEI Capability Maturity Model: Repeatable Level,* Technical Report CMU/SEI-93-TR-7, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1993.

[**Paulk 93**] Paulk, Mark C., et al., *Key Practices of the Capability Maturity Model for Software, Version 1.1*, Technical Report CMU/SEI-93-TR-25, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., 1993.

[**Paulk 95**] Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrissis, *The Capability maturity Model: Guidelines for Improving the Software Process,* Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., Oct. 1995.

[**Pence 93**] Pence, J. L. Pete and Samuel E. Hon III, "Building Software Quality into Telecommunications Network Systems," *Quality Progress*, Bellcore, Piscataway, N.J., pp. 95-97, Oct. 1993.

[**Pitts 97**] Pitts, David R., "Metrics: Problem Solved?", *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, Dec. 1997

[**Platinum 98**] © 1995, 1998 PLATINUM technology, inc. All rights reserved. 1-800-442-6861, 630-620-5000, Fax: 630-691-0718, www.platinum.com.

[**Rader 93**] Rader, Jack, Ed. J. Morris, and Alan W. Brown, *An Investigation into the State-of-the-Practice of CASE Tool Integration,* Technical Report CMU/SEI-93, Software Engineering Institute, Carnegie Mellon University, Pittsburg, Pa., 1993.

[**Schamp 95**] Schamp, Alan, "CM-Tool Evaluation and Selection," IEEE Software, 1995.

[**Semiatin 94**] Semiatin, William J., "Evolution of Configuration Management," Program Manager: Journal of the Defense Systems Management College, November/December 1994.

[**Slomer 92**] Slomer, Howard M. and Alan M. Christie, *Analysis of a Software Maintenance System: A Case Study,* Technical Report CMU/SEI-92-TR-3, ESC-TR-92-031, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., Nov. 1992.

[**Smith 93**] Smith, Dennis, et al., *Software Engineering Environment Evaluation Issues,* Technical Report CMU/SEI-93, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., March 1993.

[**Softool 92**] Softool Corporation, *Successful Software Strategies Seminar Series: Improving Your Configuration Management Implementation Strategy,* Washington, D.C., 1992.

[**Starbuck 97**] Starbuck, Ronald A., "Software Configuration Management: Don't Buy a Tool First," *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, Nov. 1997.

[**Starrett 98**] Starrett, Elizabeth C. L., "Measurement 101," *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, Aug. 1998.

[**STSC 94**] Software Technology Support Center, *Software Configuration Management Technology Report,* Software Technology Support Center, Hill Air Force Base, UT, Sept. 1994.

[**Ventimiglia 97**] Ventimiglia, Bob, *Advanced Effective Software Configuration Management*, Technology Training Corporation, 1997.

[**Ventimiglia 98**] Ventimiglia, Bob, "Effective Software Configuration Management," *CrossTalk*, *The Defense Journal of Software Engineering*, Software Technology Support Center, Hill Air Force Base, UT, Feb. 1998.

[**Wallnau 92**] Wallnau, Kurt C., *Issues and Techniques of* CASE *Integration with Configuration Management,* Technical Report CMU/SEI-92-TR-5, ESD-TR-92-5, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pa., March 1992.

[**Whitgift 91**] Whitgift, David, *Methods and Tools for Software or Software Configuration*, John Wiley & Sons Inc., New York, 1991.

[**Wreden 94**] Wreden, Nick, "Configuration Management: Getting with the Program," *Beyond Computing*, November/December 1994.

Software Technology Support Center
OO-ALC/TISE
7278 4th Street
Hill AFB, Utah 84056
801-775-5555
801-777-8069
www.stsc.hill.af.mil