

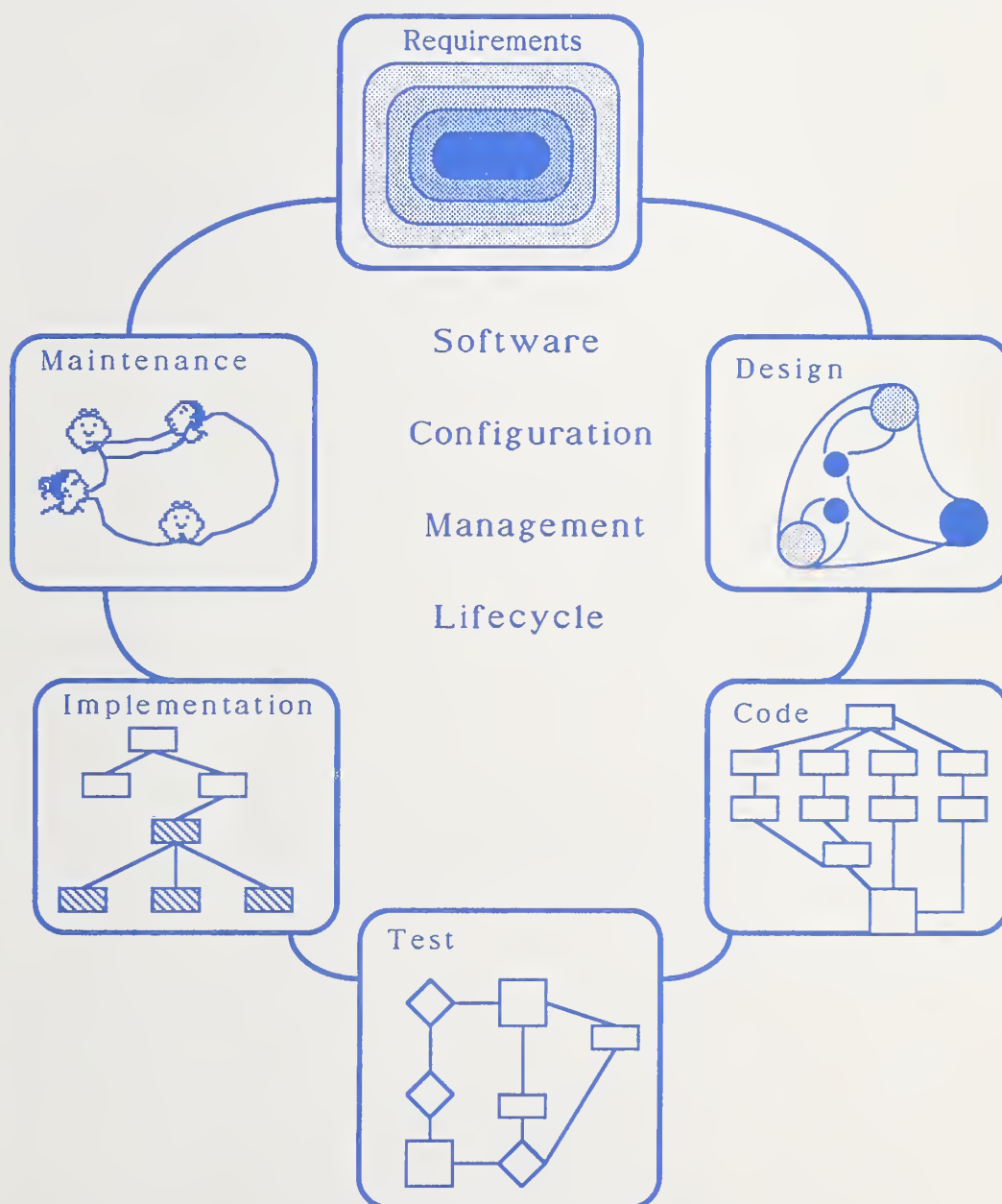


# Computer Systems Technology

NIST Special Publication 500-161

## Software Configuration Management: An Overview

Wilma M. Osborne



**NATIONAL INSTITUTE OF STANDARDS &  
TECHNOLOGY  
Research Information Center  
Gaithersburg, MD 20899**

# Computer Systems Technology

---

NIST Special Publication 500-161

## Software Configuration Management: An Overview

Wilma M. Osborne

National Computer Systems Laboratory  
National Institute of Standards and Technology  
Gaithersburg, MD 20899

March 1989



**NOTE:** As of 23 August 1988, the National Bureau of Standards (NBS) became the National Institute of Standards and Technology (NIST) when President Reagan signed into law the Omnibus Trade and Competitiveness Act.

### U.S. DEPARTMENT OF COMMERCE

Robert A. Mosbacher, Secretary

Ernest Ambler, Acting Under Secretary for Technology

### National Institute of Standards and Technology

Raymond G. Kammer, Acting Director

**Library of Congress Catalog Card Number: 89-600728**  
**National Institute of Standards and Technology Special Publication 500-161**  
**Natl. Inst. Stand. Technol. Spec. Publ. 500-161, 33 pages (March 1989)**  
**CODEN: NSPUE2**

**U.S. GOVERNMENT PRINTING OFFICE**  
**WASHINGTON: 1989**

---

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, DC 20402

## ABSTRACT

Today as software systems become more complex, it is essential that system components be identifiable, traceable, reuseable, and maintainable. The more these characteristics are present, the more likely the end products will satisfy user requirements. However, in order to produce such products, there must be consistency in the management and control of software changes. These efforts are made more difficult by the frequency with which software changes are requested, approved, and incorporated into production systems, without the aid of either formalized or automated change control. Software configuration management (SCM) provides a discipline for planning and implementing change control throughout the software lifecycle. While SCM is not an end in itself, and its use does not assure the success of a project, it is a powerful tool for providing management with greater visibility into the software process, thereby enabling management to make the "right" decisions and to deliver correct products on time and within budget. This Guide presents an overview of the software configuration management discipline, and identifies how SCM helps to improve and control lifecycle software change management.

## KEYWORDS

change; change control; CI; CPCI; CPM; CSCI; environment; libraries; SCM plan; software; software configuration management; software management; software tools; tools.

## TABLE OF CONTENTS

1.0	BACKGROUND.....	1
1.1	Introduction.....	1
1.2	Purpose.....	2
1.3	Organization.....	2
1.4	Definition of SCM.....	2
1.5	The Goals of SCM.....	3
1.6	Benefits of SCM .....	4
1.7	Levels of SCM.....	5
2.0	CHANGE MANAGEMENT AND CONTROL ISSUES.....	5
2.1	Change Management Issues.....	5
2.2	Control Issues.....	6
3.0	SOFTWARE CONFIGURATION MANAGEMENT.....	7
3.1	Configuration Identification.....	7
3.2	Configuration Control.....	11
3.2.1	Change requests.....	11
3.2.2	Change evaluation.....	11
3.2.3	Change approval/disapproval.....	12
3.2.4	Change implementation.....	12
3.3	Configuration Status Accounting.....	12
3.4	Configuration Audits and Reviews.....	13
4.0	THE ROLE OF SCM TOOLS.....	14
5.0	USING SCM LIBRARIES.....	16
6.0	SCM STANDARDS AND GUIDELINES.....	18
7.0	THE SCM PLAN.....	20
8.0	SUMMARY.....	22
	GLOSSARY.....	23
	REFERENCES.....	26
	RELATED MATERIAL.....	27
	FIGURES	
	FIGURE 1 - SW LIFECYCLE PRODUCT ASSURANCE PROCESSES.....	4
	FIGURE 2 - SCM FUNCTIONS.....	7
	FIGURE 2A- SCM FUNCTIONS AND ACTIVITIES.....	8
	FIGURE 3 - BASIC TOOL SET.....	14
	FIGURE 4 - ADVANCED TOOL SET.....	15
	FIGURE 5 - ON-LINE TOOL SET.....	15
	FIGURE 6 - LIBRARY CONTROL.....	17
	FIGURE 7 - STANDARDS AND GUIDELINES: WHERE TO LOOK.....	19
	TABLES	
	TABLE 1 - HIERARCHICAL STRUCTURE OF CONFIGURATION ITEMS.....	9



## 1.0 BACKGROUND

The National Computer Systems Laboratory (NCSL) of the National Institute of Standards and Technology (NIST), has a responsibility under Public Law 89-306 (Brooks Act), as amended by the Computer Security Act (P.L. 100-235) to promote cost effective selection, acquisition, and utilization of computer systems within the Federal Government. NCSL efforts include research, technical assistance, and the development of standards and guidelines for computer and related telecommunications systems. The NCSL is developing software configuration management(SCM) guidance designed to assist Federal agencies in improving software quality, as well as controlling the cost of software development and management.

This Guide provides an overview of software configuration management, a support function dedicated to making both the technical and managerial software activities more effective. It addresses the problems associated with managing software changes; the importance of implementing SCM procedures early; and the application of those procedures throughout the software lifecycle. A brief summary of SCM tools and their applicable functionality is provided. SCM extends to more than just the code (source, relocatable, executable) and documentation (e.g., system and software requirements and design specifications). It also covers control files, test data, test suites, support tools, and other components used to develop and maintain the software product.

### 1.1 Introduction

Today as software systems become more complex, it is essential that system components be identifiable, traceable, reusable, and maintainable. The more these characteristics are present, the more likely the end products will satisfy user requirements. However, in order to produce such products, there must be consistency in the management and control software changes. These efforts are made more difficult by the frequency with which software changes are requested, approved, and incorporated into production systems, without the aid of either formalized or automated change control. Software configuration management is not an end in itself, and its use does not assure the success of a project. It is, however, a powerful tool for providing management with greater visibility into the software process, thereby enabling management to make the "right" decisions and to deliver correct products on time and within budget.

## 1.2 Purpose

The purpose of this Guide is to identify issues that managers should consider when planning to implement SCM. It is intended to provide answers to the following questions:

1. What should a manager consider when planning for SCM?
2. What level of control or authority will be required for the various users under SCM?
3. What are the basic steps in formal SCM?
4. Is it possible to integrate formal SCM with existing informal project management procedures?
5. What should be considered when determining the appropriate level of SCM for an organization?
6. What SCM tools are appropriate for a specific environment?

## 1.3 Organization

The sections in this Guide are organized as follows:

- 2 describes issues associated with software change.
- 3 describes SCM activities.
- 4 describes SCM benefits.
- 5 describes tool sets for different environments.
- 6 describes the types and function of libraries typically utilized in SCM.
- 7 describes SCM standards and guidelines.
- 8 describes requirements for an SCM Plan.
- 9 presents a summary.

## 1.4 Definition of SCM

Software configuration management is the management of software change. The Institute of Electrical and Electronics Engineers (IEEE) Guide for Configuration Management Plans (IEEE Std. 1042-1988) defines SCM as "a formal discipline which provides methods and tools: to identify the components and baselines of a software development or maintenance effort, and to control changes to those components" [1]. The Department of Defense (DOD) MIL-STD-490 states that "Configuration management (CM) is a discipline applying technical and administrative direction and surveillance to (a) identify and document the functional and physical characteristics



of a configuration item; (b) control changes to those characteristics; and (c) record and report change processing and implementation status" [2]. Bryan and Siegel, in Software Product Assurance, Techniques For Reducing Software Risk, state that "SCM is a discipline for visibly, traceably, and formally controlling software evolution" [3].

Increasingly, organizations are recognizing that effective change control management is a key to assuring that the product delivered is indeed the product intended and expected [4,5]. SCM provides a method for logically grouping related components throughout the various phases of development, from the time a change is requested, approved, implemented, and tested, until it is released for production and into maintenance [6,7,8]. It is simply the use of a common sense approach to the complex problem of software development and management.

### 1.5 The Goals of SCM

SCM, like other aspects of product assurance, is a lifecycle discipline which should begin when the first system related document is prepared (see fig. 1). Organizations have typically considered SCM to be something akin to "bean counting" with little influence on how well intermediate or final products satisfy predefined requirements. Today, while there is still an emphasis on identification of part numbers, etc., SCM is clearly key to the product assurance program. It serves not only as a check and balance against the quality assurance process, but as the last point at which a product is determined ready or not ready for release. The goals of SCM are similar to those of product assurance. SCM is, in fact, integral to product assurance. The goals of SCM are to manage evolutionary changes to the software system responsively, and to make the entire software development process both visible and traceable. The goals of product assurance include not only the management of product development, but the responsibility for assuring that high quality, correct, reliable products are produced.

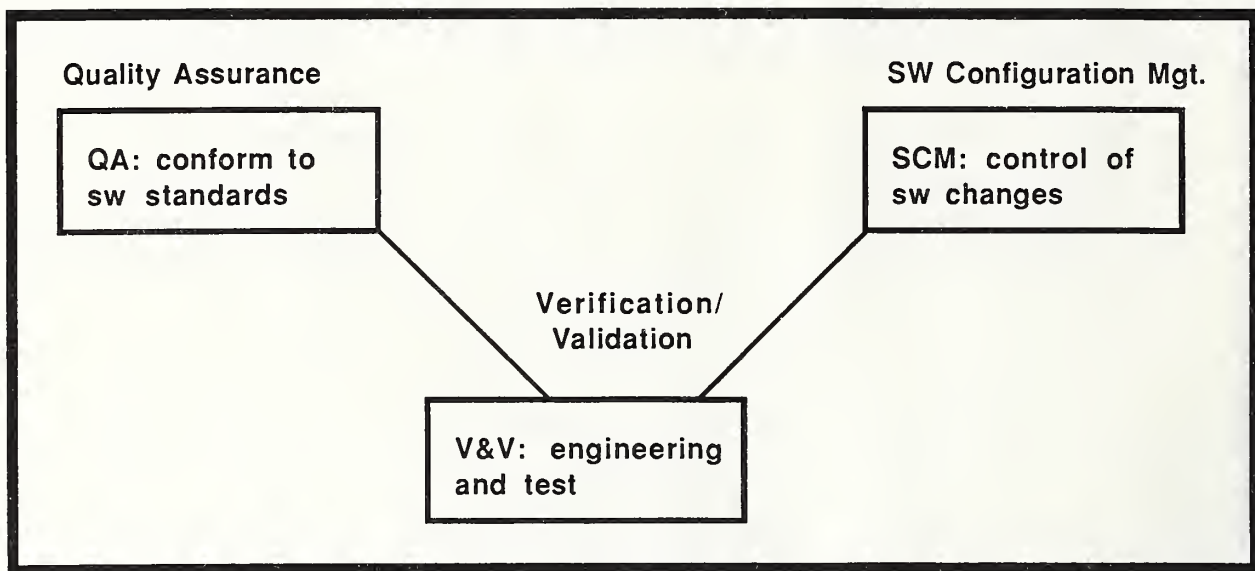


FIGURE 1. SW LIFECYCLE PRODUCT ASSURANCE PROCESSES

### 1.6 Benefits of SCM

SCM is needed by anyone involved in, or responsible for, software development and maintenance. The entire organization benefits from effective configuration management of the software systems. Management at all levels needs accurate and timely information to determine the impact of a proposed change; to be aware of the status of changes, and to make the right decisions. An effective SCM program helps to assure persons responsible for developing, modifying, and managing software, as well as users of the software, that the correct version and baseline are released for operations. It also helps to ensure that two or more persons do not work on the same file at the same time, and keeps team members aware of what other changes have been made or are being made to the software.

As software systems grow and become more complex, so does the task of managing the associated problems of change. And as more persons are needed to accomplish the tasks, and more procedures are required to ensure proper coordination and approval, there is a corresponding need for SCM. Perhaps one of the greatest advantages provided to management by SCM is the formalized control over access and change to the systems.

Another benefit of SCM is the formalized structure for identifying documentation, interfaces, software, and databases to support all lifecycle phases and disciplines. If SCM is consistently applied, it reinforces the chosen development and maintenance methodology that fits the requirements, standards, policies, and organization and management philosophy [9]. Finally, SCM, through its status accounting functions provides management and product information concerning the status of baselines, change control, tests, releases, and audits.

## 1.7 Levels of SCM

The level of formality of a SCM implementation depends on the structure, resources, and needs of the organization. The key to effectively managing software change is the exercise of clearly defined levels of control by both management and technical personnel. The authority for allocation of resources, scheduling, software change, release of new baselines, and other decisions related to the change control should be known and followed. Depending on the existing environment and procedures, it may be practical to retain the existing structure making only those changes that will help to assure effective change control management. For example, automated SCM tools may be essential to ensure the level of tracking, traceability, and control required for air traffic control systems, banking systems, space station systems, payroll, and other large systems. Such tools may not be required for small, less critical systems or in a small organization where SCM is performed by the same person or group who identifies the need for change, performs and tests the change, and approves the new baseline. More important is that persons who will have responsibility for implementing SCM are involved in the initial planning for SCM, and are not overburdened with SCM procedures.

## 2.0 CHANGE MANAGEMENT AND CONTROL ISSUES

### 2.1 Change Management Issues

Anytime it is possible for two persons to simultaneously modify a software system, there is a change control problem. If someone changes the code, but not the associated user manuals and documentation, leaving subsequent users to find out the hard way, there is a breakdown in the change management. Neither of these incidents should occur in an environment with effective change management. Barry Boehm, in Software Economics [5], estimates the cost of documentation to be 50% of the software development costs. Numerous studies including those conducted as part of the NIST software maintenance project [10] have concluded that the cost of incorrect and obsolete documentation accounts for a significant portion of the software dollar. Such documentation can result in improperly functioning software, failed missions, and lost time. In the case of critical software systems where such breakdowns in change control could result in loss of life, an effective change management system is essential. Thus, the cost of not having adequate software change management and control is too high. Cost is cited here as one of the primary issues because that is the bottom line whether the systems in question support defense, transportation, education, health, insurance, food production, or any other business. Software change is an evolutionary process integral to the growth and



effectiveness of most systems. Everything that happens to a software system as it evolves is dependent upon the access or privilege to change it and the extent that the documentation of the system reflects the intent, functionality, and use of the system.

Other change management issues of concern include:

- o lack of visibility into what is occurring
- o lack of traceability of software changes to requirements specifications
- o ad hoc changes in the absence of impact analysis.

## 2.2 Control Issues

Everything that is used to produce the end product is to some extent important. Recognizing that it would be impossible to control such a large volume of information, it is essential to identify and assign a name and version/release number to the requirements and design documents, source code, control language, tools, and any documentation considered useful to the development, understanding, and use of the system. A unique identifier makes it possible to control the various library versions and baselines. Some examples of what should be controlled by SCM through the use of version numbers or other identification include:

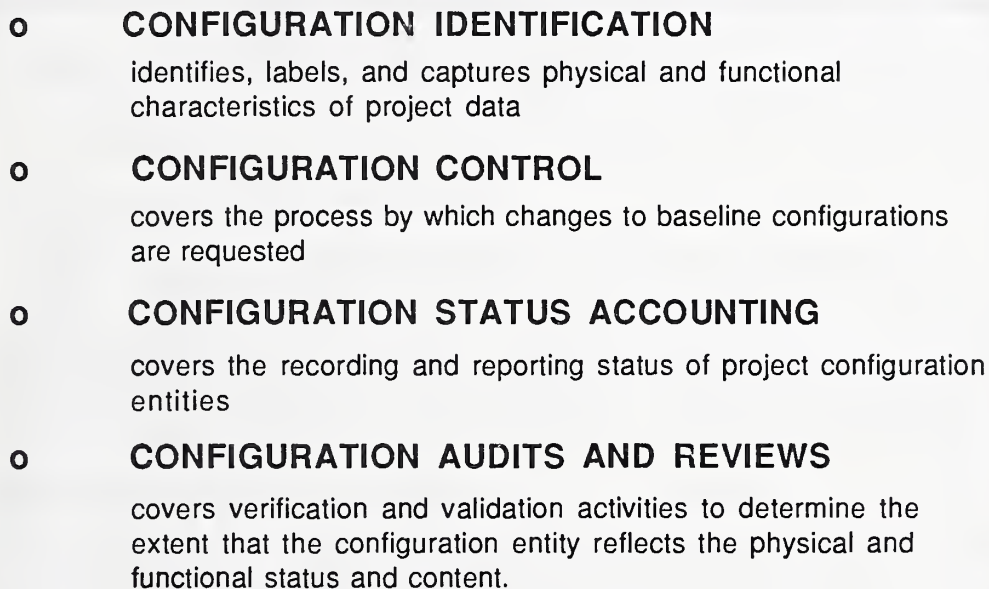
- o project management plans
- o product assurance plan
- o software management plan
- o requirements specifications
- o design plans and documents
- o test plans, designs, data, cases, and procedures
- o software maintenance plan
- o source code, relocatable code, executable code
- o common blocks
- o prototype (baseline/final version)
- o control language
- o libraries
- o tools
- o hardware configuration
- o firmware
- o other system information as determined.

Prototyping is becoming an increasingly important approach for software development. Prototyping is a low cost, quick method of iterating to a solution. It is unnecessary to baseline each prototype since this would not only constrain flexibility, but could increase product development costs. However, once the

prototype satisfies the user requirements, it is often accepted as a deliverable and becomes part of the system. Whenever such decisions are made, whether the prototype is a design document or a software product, the prototype must be placed under configuration management.

### 3.0 SOFTWARE CONFIGURATION MANAGEMENT

The four key SCM functions are configuration identification, configuration audits and reviews, configuration status accounting, and configuration control (see figs. 2 & 2A.). While each of these functions is important, the configuration identification of software components is the most critical for effective SCM.

- 
- o **CONFIGURATION IDENTIFICATION**  
identifies, labels, and captures physical and functional characteristics of project data
  - o **CONFIGURATION CONTROL**  
covers the process by which changes to baseline configurations are requested
  - o **CONFIGURATION STATUS ACCOUNTING**  
covers the recording and reporting status of project configuration entities
  - o **CONFIGURATION AUDITS AND REVIEWS**  
covers verification and validation activities to determine the extent that the configuration entity reflects the physical and functional status and content.

**FIGURE 2. SCM FUNCTIONS**

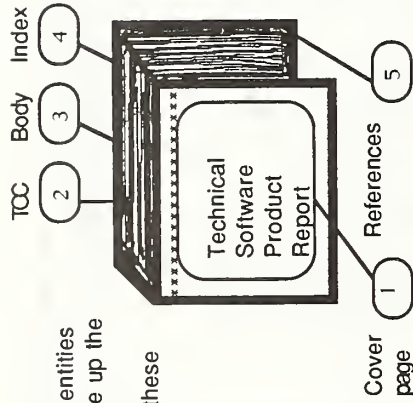
#### 3.1 Configuration Identification

Configuration Identification activities identify, label, and capture the physical and functional characteristics of project configuration entities and baselines (e.g., documentation, systems, program versions, modules). The configuration entities



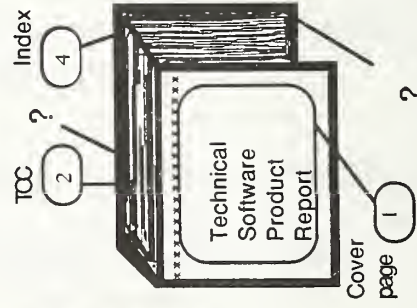
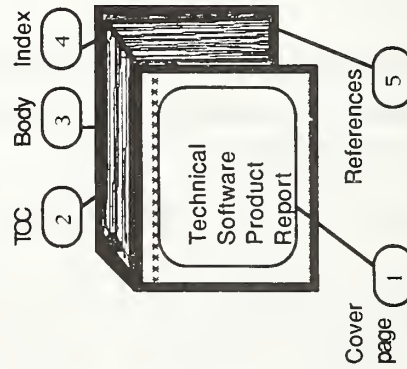
## IDENTIFICATION

- Labeling and recording product entities
- o what are the entities that make up the product
  - o what are the relationships of these entities to each other



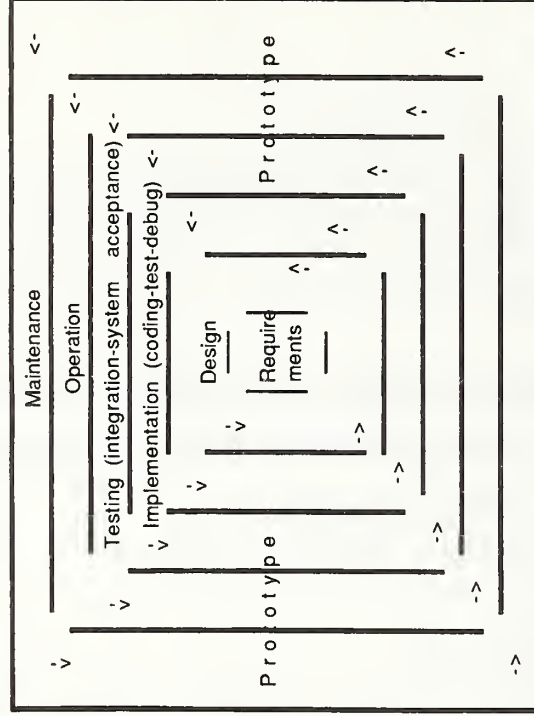
## CONTROL:

- Change only what has been approved
- o impact- of changes on existing or new software system
  - o decision- to make changes
  - o responsibility- for making changes



## AUDIT:

- Ensure that product satisfies requirements
- o have contractors/organizations performed all work necessary to meet stated requirements
  - o was the product audited as it evolved to prevent unanticipated problems later



## STATUS ACCOUNTING:

- Monitoring the change activity
- o what happened
  - o when did it happen

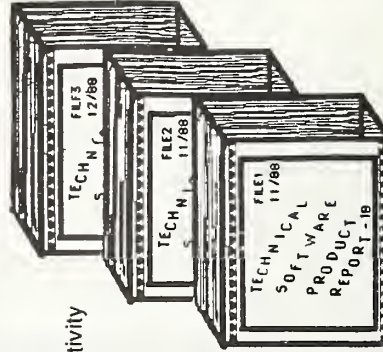


FIGURE 2A. SCM FUNCTIONS AND ACTIVITIES

are defined in terms of the hierarchical structure of software elements which comprise the project configuration items and the software elements which are based on required levels of control. An example of a hierarchical structure of a configuration item is shown below.

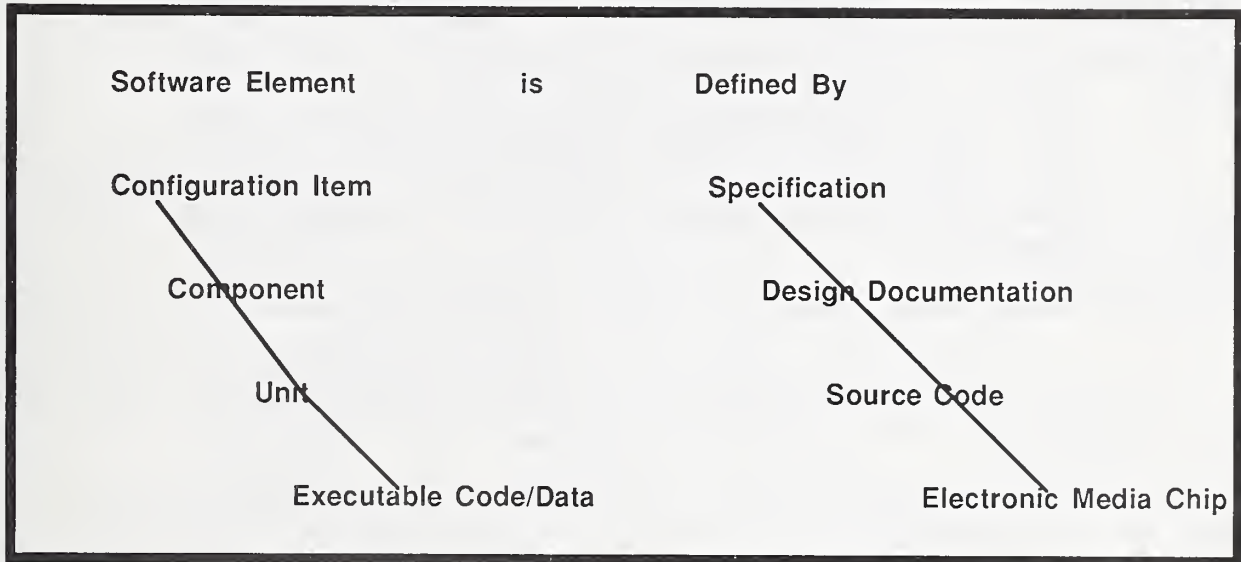


TABLE 1. HIERARCHICAL STRUCTURE OF CONFIGURATION ITEMS

Baselines refer to a specification or product: that has been accepted by the responsible management; that will serve as the basis for further development; and that can be changed only through formal change procedures. A baseline represents the assignment of a unique identifier to each configuration item (CI), computer program configuration item (CPCI), or computer software configuration item (CSCI), and the associated entities. (See glossary for further discussion.) Baselines provide a means of releasing (internally or externally) each CI, CPCI, or CSCI from one phase of development to another. The associated entities include the software designs, source code, relocatable code, executable code, engineering changes, files used in the process of executable code, documentation, and tools used to support software development and maintenance. The baselines are also used to control these entities during various stages of the software development process.

The three most commonly used baselines are the functional, allocated, and product baselines, although an organization may specify others as appropriate. The functional baseline is generally referred to as the one approved by the user or customer after a review of the system/software requirements.

The allocated baseline is generally established after the user or customer reviews the specifications and requirements documentation. Before the software product can be released, it must pass both a physical configuration audit and a functional configuration audit. The product baseline is established after the user or customer has approved the product specification following the functional configuration audit. All of these baselines must be approved by the user or customer in coordination with the developer or other approval authority.

Baselines should be defined in terms of:

1. The event that creates the baselines.
2. The entities that are controlled.
3. The procedures used to establish and change a baseline.
4. The authority required to approve changes to a baseline.

Classes of entities that are to be managed include:

1. Configuration items and dependent software elements to be newly developed. They should be given unique identification schemes.
2. Configuration items and dependent software elements to be modified. They should be given identification schemes compatible with current usage. Version and revision labeling is an example of re-identification techniques.
3. Third party software to be identified. It should be identified in a manner consistent with configuration items to be incorporated. Vendor proprietary software should be uniquely identified in a configuration item to preserve its logical and legal integrity.
4. Support software to be used. This includes support software used during the development and maintenance of project configuration items.

SCM procedures should also address such activities as acquisition, library storage, retrieval, and reproduction of configuration entities as they relate to configuration management. Each is described in more detail below.

1. Acquisition. Activities to be considered include:
  - (a) Identification of vendor/subcontracted software, reused software, and system software used in support of development and maintenance activities (i.e., definition of version/revision schemes).



- (b) Inclusion of configuration elements into a controlled library (i.e., entering a unit into a controlled computer data base).
  - (c) Establishment of project baselines (i.e., preparation of version description documentation).
  - (d) Receiving and inspection activities.
2. Library Storage. The SCM procedures should describe the labeling activities associated with the physical storage of documentation and magnetic media. Steps should be taken to guard against the loss of information due to fire, theft, etc.
  3. Retrieval. The SCM procedures should describe the procedures to retrieve entities from library storage. These activities should include: verification of marking and labeling, tracking of data, and security of proprietary information where applicable.
  4. Reproduction. The SCM procedures should describe the necessary marking and labeling for reproducing and distributing a configuration item. Activities may include: version/revision marking, labeling of documentation and executable software elements, serialization and altered item marking for executable code or data embedded on a microchip, and identification of physical packaging data.

### 3.2 Configuration Control

The Configuration Control activities include the process by which changes to baseline configuration entities are requested, evaluated, approved, disapproved, and if applicable, implemented. The SCM procedures should discuss methods of controlling configuration entities residing in project libraries.

#### 3.2.1 Change requests

SCM procedures should define the specific steps taken to analyze and evaluate the change request, clarify the meaning of the request, and resolve the problem described.

#### 3.2.2 Change evaluation

SCM procedures should identify the appropriate individuals or organization responsible for evaluating the change requests. The procedures should also discuss the procedures for submission of

the evaluation results to the appropriate review board or individuals for approval/disapproval.

### 3.2.3 Change approval/disapproval

SCM procedures should identify the individuals or organizations with the authority to approve change requests. The level of authority may be predicated upon the degree of system complexity and impact. However, during the system lifecycle, the approval authority may change several times. If project review boards are established to have approval authority, the plan should indicate for each board: (a) the period it will be in effect, (b) its membership, (c) its voting and veto powers, and (d) if applicable, its relationship to other project review boards.

### 3.2.4 Change implementation

Care must be taken to insure that persons who will have responsibility for implementing SCM are involved in the initial planning for SCM, and are not overburdened with SCM procedures. The information associated with implementation of the change should contain as a minimum the:

- (a) referenced change request number
- (b) date implemented
- (c) date verified
- (d) person or organization responsible for verification
- (e) person or organization responsible for installing change
- (f) components/units of the CI affected (if applicable)

Additional information related to the change implementation, such as the identification of the supporting software used to implement the change, may be included.

## 3.3 Configuration Status Accounting

Configuration Status Accounting activities are associated with the recording and reporting of the status of project configuration entities throughout the system lifecycle. Data elements to be tracked and reported include the initially approved configuration of an entity, the status of requested changes to the entity, and the implementation status of approved changes. The level of detail required may vary according to the project or customer's information needs.

The following are examples of the type of questions that should be addressed:

1. What kind of data elements are to be tracked and reported?



2. How is information about the configuration entities to be collected, verified, stored, processed and reported?
3. What types of status accounting reports are to be generated and how frequently?
4. Who is responsible for the tracking, controlling and reporting of data?

### 3.4 Configuration Audits and Reviews

Configuration audits are verification and validation activities that determine to what extent the configuration entity reflects its actual physical and functional status and content. The functional audit is performed on the configuration item after acceptance tests have been completed. The physical audit of the configuration item is performed to verify that the final listings and the final version of the specifications conform to the built configuration item. Configuration audits and reviews are planned for any release of a baseline that elevates the level of authority required to approve changes to the new baseline. Both the user or customer and others as designated, participate in the audits and reviews [1,3,4].

The management of reviews and audits should include:

1. Requirements and procedures for conduct of meetings
2. The list of participants, schedules and organizational responsibilities
3. Documentation which is to be supplied at each review or audit
4. The process for approval, corrective action, and follow-up.

The reviews that have been scheduled or that are required should be described. The objectives of each review should be defined and the baseline to be established or transferred should be listed by its prescribed title, e.g., Product Baseline.

The audits that have been scheduled or that are required should be defined. Such audits will either verify that the functional software requirements have been tested successfully as reflected in the resulting test data or that the physical software is described in the documentation to be delivered with the software. Together, these functions help to control the interim, as well as completed, versions of the software system.

#### 4.0 THE ROLE OF SCM TOOLS

There are numerous SCM tools available which have a wide range of capabilities. Some do little more than version control, while others keep track of every access, change, and release. Others can identify when files should be replaced, deleted, or merged, and can facilitate comparison of versions for differences. Still others support a methodology that permits linking the physical change document to the change as it is incorporated in the system. Many of the SCM tools are bundled as part of the operating system. A number of these tools can be put into use after only a few days of training. Some firms provide SCM tools as part of their support service or maintenance agreements.

One way to classify tools is according to the characteristics of the products for which they will be used. Another way is to examine the functions they perform. Finally, tools may be classified according to how they fit into the software engineering environment.

There are several classifications of tool sets found in SCM environments. The three most common are a basic tool set, an advanced tool set, and an on-line tool set. As indicated by the figures below, the set depicted in figure 4 is more comprehensive and provides more capability than the one in figure 3; while the set depicted in figure 5 provides more capability than the set in figure 4. The selection of tools used to aid software change management, however, will depend on the specific environment and the requirements of the organization.

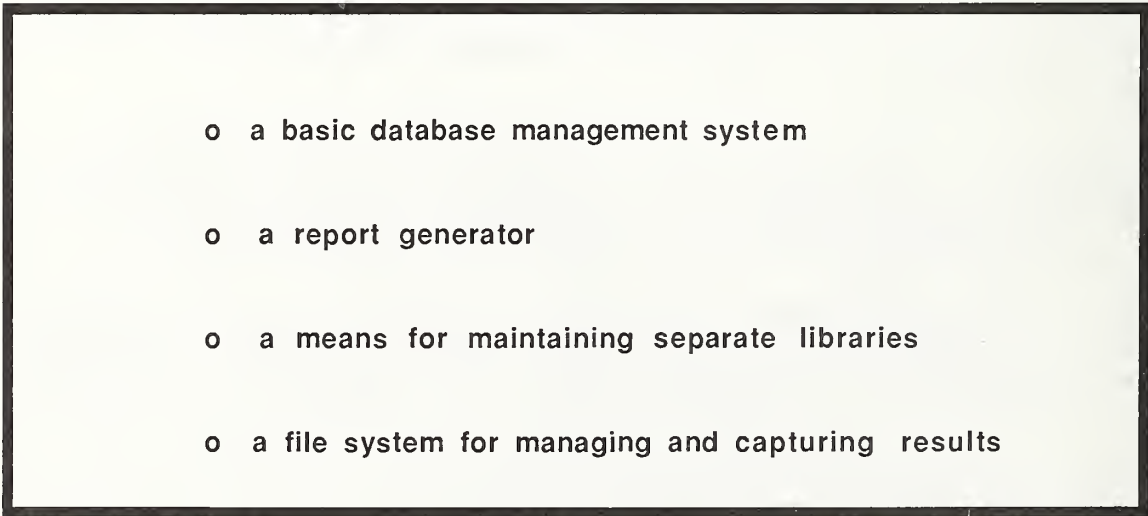
- 
- o a basic database management system
  - o a report generator
  - o a means for maintaining separate libraries
  - o a file system for managing and capturing results

FIGURE 3. BASIC TOOL SET

- o items in the basic tool set
- o source code control programs for version control
- o compare programs for verifying changes
- o tools for building executable code
- o documentation/word processing system to capture and maintain files
- o change request/authorization system for tracking change requests in machine readable form

FIGURE 4. ADVANCED TOOL SET

- o generic tools of the advanced tool set integrated so that they can work from a common database
- o system/software change request/authorization(SCR/SCA) tracking system that permits generations, reviews, and approval of changes on-line
- o report generators which use common database and handle on-line queries

FIGURE 5. ON-LINE TOOL SET

## 5.0 USING SCM LIBRARIES

SCM libraries are generally composed of hard copy and software on machine readable media. The libraries are an integral part of the software engineering environment. They provide the means for identifying and labeling baseline entities and for capturing and tracking the status of changes.

The number and kinds of libraries will vary from project to project according to variations in the access rights and needs of their users. However, there are fundamentally three kinds of libraries for the controlled collection of configuration entities associated with defined baselines (see fig. 6). They are:

- |                    |  |
|--------------------|--|
| dynamic library    | This library is sometimes referred to as the "programmers library" used for newly created or modified software elements. It is used for developing code and is freely accessible to the programmer.  |
| controlled library | This library is sometimes called the "master library" and is used for managing current baselines and controlling changes to them. This is the library where components of configuration items that are ready to be integrated are maintained. Copies may be made available to programmers and others, but use of this library must be authorized by a delegated authority. |
| static library     | This library is the "software repository" and is for general use baselines. This is where the master copies of the computer program configuration items are maintained.  |

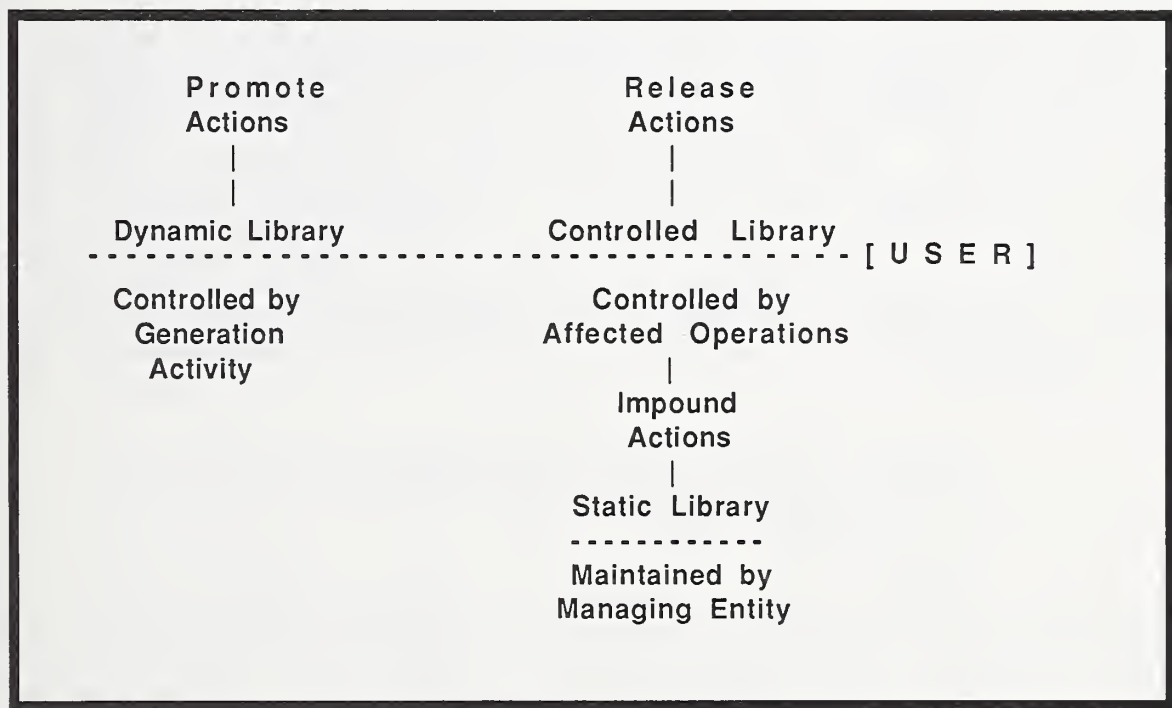


FIGURE 6. LIBRARY CONTROL



## 6.0 SCM STANDARDS AND GUIDELINES

The Department of Defense (DOD) has a number of SCM standards for the various services; the MIL-STD-490 is the most widely used [2]. However, DOD Std. 2167A which was published in 1987 and revised in 1988 [11], provides a data item description (DID) for SCM functions and activities. The Office of the Secretary of Defense is currently conducting a survey of SCM guidance, and is compiling a list of the existing Federal SCM standards and guidelines with the intent of eliminating redundancy. While most organizations have and employ some formal or informal change control procedures, many software managers agree that comprehensive Federal and industry SCM standards and guidelines are needed [4].

In 1982, the IEEE Computer Society's Standards Board established a Working Group for the development of a Standard for SCM Plans. The first standard was published in 1983 (IEEE Std. 828-1983). The Guide for Developing SCM Plans was completed in 1988 (IEEE Std. 1042-1988). This Guide contains two key features: SCM plans for four different environments, and a "Things to Consider" section after each major discussion throughout the document. The four SCM plans are based on actual plans used in production environments. The purpose for including actual plans was to provide additional guidance for individuals faced with the responsibility of implementing a SCM program. Areas addressed by the plans are:

- 1 - a large software development,
- 2 - a mission critical software effort,
- 3 - a software maintenance effort, and
- 4 - a small product line.

The IEEE Computer Society Standards' policy requires that all standards be revisited and revised as appropriate every 5 years. Work on the 828-1983 SCM Standard is underway. The revised SCM standard should be published in early 1989. An IEEE Tutorial on SCM published in 1985 provides a comprehensive review of technical SCM papers and case studies from industry and government. The Software Engineering Institute conducted a workshop on software configuration management in 1987. The proceedings from that workshop may be obtained from the address listed in figure 7.

ORGANIZATION	INDEX
1. American National Standards Institute 1430 Broadway New York, NY 10018	Catalog
2. Electronic Industries Association 2001 Eye St. NW Washington, DC 20006	Catalog
3. IEEE Computer Society Service Center 445 Hoes Lane Piscataway, NJ 08854	Catalog
4. National Technical Information Service 5285 Port Royal Road Springfield, VA 22161	NIST/FIPS INDEX
5. Superintendent of Documents U.S. Government Printing Office Washington, DC 20402	DOD INDEX
6. Naval Publications and Forms Center 5801 Tabor Avenue Philadelphia, PA 19120	Individual DOD Standards
7. International Standards Organization Case Postale 256, CHI-1211 Geneva 20, Switzerland	Catalog
8. Software Engineering Institute Carnegie-Mellon University Pittsburgh, PA 15213	Technical Reports

FIGURE 7. STANDARDS AND GUIDELINES: WHERE TO LOOK

## 7.0 THE SCM PLAN

The SCM Plan (SCMP) is the blueprint for implementing SCM effectively. It documents what SCM is to be done, how it is to be done (including how it fits with the supported activity), who is responsible and when it is to happen. In essence, the plan should address the SCM operational concept over the product lifecycle; the identification, traceability, control, audit, review and transfer of the specified baselines; and the SCM process itself. The plan is a living document which should be updated to keep current with the product lifecycle phase and the project management approach. The structure and content of an SCM Plan may vary, but there are some basic criteria that must be included. As a minimum, the SCM plan should contain the following:

- |                        |   |
|------------------------|---|
| o Introduction of Plan | Describes the plan's purpose, scope of application, and key terms.                        |
| o SCM Management       | Identifies the responsibilities and authorities for accomplishing the planned activities. |
| o SCM Activities       | Identifies all project activities to be performed in applying SCM.                        |
| o SCM Schedules        | Identifies required coordination of SCM activities with the overall project schedule.     |
| o SCM Resources        | Identifies tools, and physical and human resources required for plan execution.           |
| o SCM Plan Maintenance | Identifies how SCM plan information will be kept current while in use.                    |
| o SCMP Appendices      | Identifies attachments which may supplement the SCMP.                                     |

The Introduction of the plan provides a synopsis of the essential SCM functions for the project. It contains an overview of the project SCM activities to be described in the plan. Software configuration management plans take many different forms. They range from stand-alone, fully enriched documents to a single section of another project document (e.g., project plan, software development plan) that references other sections of the project document for actual detailed content.

At a minimum, introductory information should address three topics: the purpose of the plan, the scope, and definition of key terms and phrases. Terminology utilized should be familiar to the reading audience and should simplify the discussion of SCM activities so that those approving, those performing, and those interfacing with the SCM activities obtain a clear and basic understanding of what the document offers. The introduction should be as brief as possible yet provide adequate understanding without requiring the reader to review the complete set of SCM planning documentation.

The purpose should provide the following information:

- o Where the SCM planning information is located
- o Overview description of the program or project
- o Identification of other software and/or project data to be included as part of the plan (e.g., support or test software)
- o Establishment of limitations such as time constraints that may apply to the SCM plan
- o All organizational units which participate in or are responsible for any SCM activity on this project
- o Interface relationships between organization units
- o Organizational charts, supplemented by statements of function and interfaces.

The sections under management should address responsibilities, activities, schedules, resources, and maintenance of the plan. The allocation of SCM activities to organizational units should be concisely described, and for each activity listed in the "SCM Activities" section, the name of the organization or individual



performing this activity should be provided. A matrix that relates the organization units to the SCM functions, activities, and tasks can be used to document the SCM responsibilities.

## 8.0 SUMMARY

The larger and more complex the project, the greater the need for unanticipated, as well as planned changes. Under these conditions, communications between change requestors, change implementors, and change managers can often become convoluted and misinterpreted. The prohibitive cost of software changes made under these conditions has forced organizations to examine the software management process for areas that can be improved. Managing and controlling software change has been identified as an area in need of improvement. SCM, one of the key components of product assurance, can help to assure that change occurs in an orderly and controlled manner, and that the software products satisfy the stated requirements. When consistently enforced, it is a tool which management can use to verify that the baselines of a software development and maintenance effort reflect the actual status of the system and its associated products.

SCM can effectively control changes to those baselines and help to assure system integrity and traceability throughout the software lifecycle by providing a foundation for product and performance measurement. SCM accomplishes this by providing the means to: identify the software developed; establish baselines; control changes to these baselines; record and track change status; and support the auditing of controlled systems.

SCM is a discipline that can be introduced into an existing environment. It is important, however, that SCM concepts are integrated into existing procedures, as opposed to just adding on a new set of procedures. SCM is the means through which the continuity of the software system is recorded, communicated, and controlled. Implementation of a SCM program can help to place control of the software system where it should be, while providing the appropriate levels of access to those who need to use it.



## GLOSSARY

**Component.** This is also known as Computer Program Component/CPC, Computer Software Component/CSC, Subsystem, Unit, Package, Program. Components are collections of units or other components. Multiple component levels may exist within a configuration item. [\*]

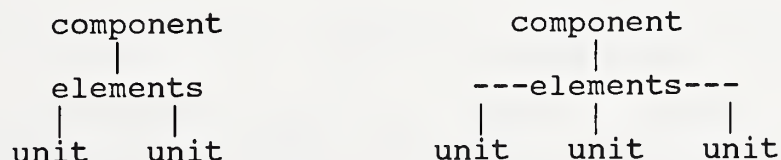
**Configuration control.** The process of evaluating, approving or disapproving, and coordinating changes to baseline configuration items or their entities.[\*]

**Configuration entities.** All the outputs of a software project, both intermediate and final, and the elements of the project support environment which are to be managed by SCM disciplines during the lifecycle of development and maintenance. Examples of intermediate outputs are management plans, specifications, test cases, and test plans. Examples of final output are source code, executable code, user documentation, databases, and program listings. Examples of support environment are compilers, operating systems, programming tools, SCM tools, and test beds.[\*]

**Configuration identification.** (1) The systematic process of designating the configuration entities in a system and recording the characteristics, both physical (marking and labeling) and functional (engineering documentation), of a configuration item and its entities. (2) The approved documentation that defines configuration items.[\*]

**Configuration Item.** This is also, referred to as Computer Software Configuration Item (CSCI), Computer Program Configuration Item (CPCI), System, System Segment, and Program. The final output of a software project, composed of a hierarchy of configuration items can be structured into a hierarchy of control levels referred to as elements. Elements can be further separated into components and elements as in the example below.[\*]

### CONFIGURATION ITEM



[\*] - Adapted from IEEE Standards Glossary of Software Terminology (IEEE Std. 729) and IEEE Guide to Software Configuration Management Plans (IEEE Std. 1042) for consistency.

**Configuration validation.** The process of validating that an identified configuration fulfills the system/software function to be performed at each chosen milestone point.[\*]

**Configuration verification.** The process of verifying that a CI and/or its hierarchical elements and associated entities are correct and complete for a stated baseline; verifying that identified system/software configurations are what they were intended to be and proclaimed to be.[\*]

**Elements.** The parts of a software program which have been designated as control levels for configuration management purposes. Elements make up the hierarchical structure of a configuration item's software program(s).[\*]

**Interface control.** The process of identifying all characteristics relevant to the interfacing of two or more configuration items and ensuring that proposed changes to these characteristics are evaluated and approved prior to implementation.

**Promotion.** An informal process of establishing a baseline internal to the project for a configuration item or its entities.

**Release.** The process of moving a baseline configuration item between organizations, such as from vendor to customer.

**Review board.** The authority responsible for evaluating and approving or disapproving proposed changes to a system and ensuring implementation of approved changes.

**Revision.** A change to a baseline configuration item that encompasses error correction, minor enhancements, or adaptations but to which there is no change in the functional capabilities.

**Software library.** The controlled collection of configuration entities associated with defined baselines. Libraries can be considered in three generic categories:

dynamic library - used for newly created or modified software elements

controlled library - used for managing current baselines and controlling changes to them

static library - used to archive baselines[\*]

[\*] - Adapted from IEEE Standards Glossary of Software Terminology (IEEE Std. 729) and IEEE Guide to Software Configuration Management Plans (IEEE Std. 1042) for consistency.

**Unit.** The smallest logical element in a software program to which configuration management can be effectively and efficiently applied. Unit is also referred to as a procedure, routine, and module.

**Version.** A change to a baseline configuration item that modifies its functional capabilities. As functional capabilities are added to, modified within, or deleted from a baseline configuration item, its version identifier changes.

[\*] - Adapted from IEEE Standards Glossary of Software Terminology (IEEE Std. 729) and IEEE Guide to Software Configuration Management Plans (IEEE Std. 1042) for consistency.

## REFERENCES

- [1] IEEE Guide For Software Configuration Management Plans, IEEE Standard 1042 1988, IEEE Computer Society Press, August 1988.
- [2] Military Standard MIL-STD-490, Configuration Management Practices for Systems, Munitions, and Computer Programs, December 31, 1970.
- [3] Bryan, William L. and Siegel, Stanley G., Software Product Assurance, Techniques for Reducing Software Risk, New York, NY: Elsevier, 1988.
- [4] Osborne, Wilma M., "All About Software Maintenance: 50 Questions and Answers," Journal Information Systems Management, Vol. 5, No. 3, published by Auerbach, Summer 1988.
- [5] Boehm, Barry W., Software Engineering Economics, Englewood Cliffs, NJ: Prentice-Hall, 1981.
- [6] SEI, "Support Materials for Software Configuration Management," SEI Curriculum Module SEI-SM-4-1-1, Software Engineering Institute, Carnegie Mellon University, September 1986.
- [7] SEI, "Support Materials for Software Configuration Management," SEI Support Material, SEI-SM-4-1-0, Software Engineering Institute, Carnegie Mellon University, September 1986.
- [8] Harvey, Katherine, "Summary of the Version Control: Software Configuration Management," Software Engineering Institute, Carnegie Mellon University, September 1986.
- [9] Tichy, Walter F., "RCS - A System For Version Control, Software - Practice and Experience, Vol. 15, No. 7, July 1985.
- [10] NBS-SP 106, "Guidance on Software Maintenance," 1983.
- [11] FIPS PUB 106 "Guidelines on Software Maintenance," 1984.



## RELATED MATERIAL

- [1] Bersoff, E.H., Henderson, V.D., and Siegel, S.G., Software Configuration Management: An Investment in Product Integrity, Englewood Cliffs, NJ: Prentice-Hall, 1980.
- [2] Freedman, Daniel P., and Weinberg, Gerald M., Handbook of Walkthroughs, Inspections, and Technical Reviews: Evaluating Programs, Projects, and Products, 3rd ed. Boston: Little, Brown, and Company, 1982.
- [3] "General Electric Quality Assurance Procedures and Software Configuration Management Procedures", January 1982.
- [4] NBS FIPS 132, "Standard for Software Verification and Validation Plans," 1988.
- [5] Feller, Peter H., "The Project Management Experiment," CMU/SEI-88-tr-7, ESD-TR-\*\*-008, Software Engineering Institute, Carnegie Mellon University, July 1988.
- [6] Wilburn, N.P., "Standards and Guidelines Applicable to Scientific Software Cycle." Hanford Engineering Development Laboratory, Westinghouse Hanford Company, HEDL-TC-2314, Richland, WA., January 1983.





U.S. DEPT. OF COMM. <b>BIBLIOGRAPHIC DATA SHEET</b> (See instructions)		1. PUBLICATION OR REPORT NO. NIST/SP- 500/161	2. Performing Organ. Report No.	3. Publication Date March 1989
4. TITLE AND SUBTITLE  Software Configuration Management: An Overview				
5. AUTHOR(S)  Wilma M. Osborne				
6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)  NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (formerly NATIONAL BUREAU OF STANDARDS) U.S. DEPARTMENT OF COMMERCE GAITHERSBURG, MD 20899			7. Contract/Grant No.	
			8. Type of Report & Period Covered  Final	
9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)  Same as Item 6				
10. SUPPLEMENTARY NOTES  Library of Congress Catalog Card Number: 89-600728  <input type="checkbox"/> Document describes a computer program; SF-185, FIPS Software Summary, is attached.				
11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)  Today as software systems become more complex, it is essential that system components be identifiable, traceable, reuseable, and maintainable. The more these characteristics are present, the more likely the end products will satisfy user requirements. However, in order to produce such products, there must be consistency in the management and control of software changes. These efforts are made more difficult by the frequency with which software changes are requested, approved, and incorporated into production systems, without the aid of either formalized or automated change control. Software configuration management (SCM) provides a discipline for planning and implementing change control throughout the software lifecycle. While SCM is not an end in itself, and its use does not assure the success of a project, it is a powerful tool for providing management with greater visibility into the software process, thereby enabling management to make the "right" decisions and to deliver correct products on time and within budget. This Guide presents an overview of the software configuration management discipline, and identifies how SCM helps to improve and control lifecycle software change management.				
12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons) change; change control; CI; CPCI; CPM; CSCI; environment; libraries; SCM plan; software; software configuration management; software management; software tools; tools.				
13. AVAILABILITY  <input checked="" type="checkbox"/> Unlimited  <input type="checkbox"/> For Official Distribution. Do Not Release to NTIS  <input type="checkbox"/> Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.  <input type="checkbox"/> Order From National Technical Information Service (NTIS), Springfield, VA. 22161			14. NO. OF PRINTED PAGES  33  15. Price	



**ANNOUNCEMENT OF NEW PUBLICATIONS ON  
COMPUTER SCIENCE & TECHNOLOGY**

Superintendent of Documents,  
Government Printing Office,  
Washington, DC 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Institute of Standards and Technology Special Publication 500-.

Name \_\_\_\_\_

Company \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip Code \_\_\_\_\_

(Notification key N-503)





# **NIST** *Technical Publications*

## ***Periodical***

---

**Journal of Research of the National Institute of Standards and Technology**—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

## ***Nonperiodicals***

---

**Monographs**—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

**Handbooks**—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

**Special Publications**—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

**Applied Mathematics Series**—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

**National Standard Reference Data Series**—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published quarterly for NIST by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements are available from ACS, 1155 Sixteenth St., NW., Washington, DC 20056.

**Building Science Series**—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

**Technical Notes**—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

**Voluntary Product Standards**—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program as a supplement to the activities of the private sector standardizing organizations.

**Consumer Information Series**—Practical information, based on NIST research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

*Order the above NIST publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

*Order the following NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

**Federal Information Processing Standards Publications (FIPS PUB)**—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

**NIST Interagency Reports (NISTIR)**—A special series of interim or final reports on work performed by NIST for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Service, Springfield, VA 22161, in paper copy or microfiche form.

**U.S. Department of Commerce**

National Institute of Standards and Technology

(formerly National Bureau of Standards)

Gaithersburg, MD 20899

Official Business

Penalty for Private Use \$300