Software Configuration Management (Chapter 5)

- Change is inevitable when software is built.
- *Software configuration management* (SCM) is an umbrella activity that is applied throughout the software process.
- SCM activities are developed to
  (1) identify change,
  (2) control change,
  (3) ensure that change is being properly implemented, and
  (4) report changes to others who may have an interest.

Difference between SCM and Software Support(Maintenance)

➢ Support is a set of software engineering activities that occur after software has been delivered to the customer and put into operation.

➢ Software configuration management is a set of tracking and control activities that begin when a software engineering project begins and terminate only when the software is taken out of operation.

The output of the software process are:

1) ***computer programs*** (both source level and executable forms);
2) ***documents*** that describe the computer programs (targeted at both technical practitioners and users), and
3) ***data*** (contained within the program or external to it).

➢ The items that comprise all information produced as part of the software process are collectively called a *software configuration.*

➢ As the software process progresses, the number of software configuration items (SCIs) grows rapidly. A hierarchy of SCIs is created as each SCI creates new SCIs.. There will be little confusion if any SCI are involved but there is always a factor called ***Change*** that generates a lot of confusion.

➢ Change may occur at anytime, for any reason.

➢ First Law of System Engineering states: "*No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the life cycle.*"

❖ **Fundamental Sources of Change**

➢ *New business or market conditions* dictate changes in product requirements or business rules.

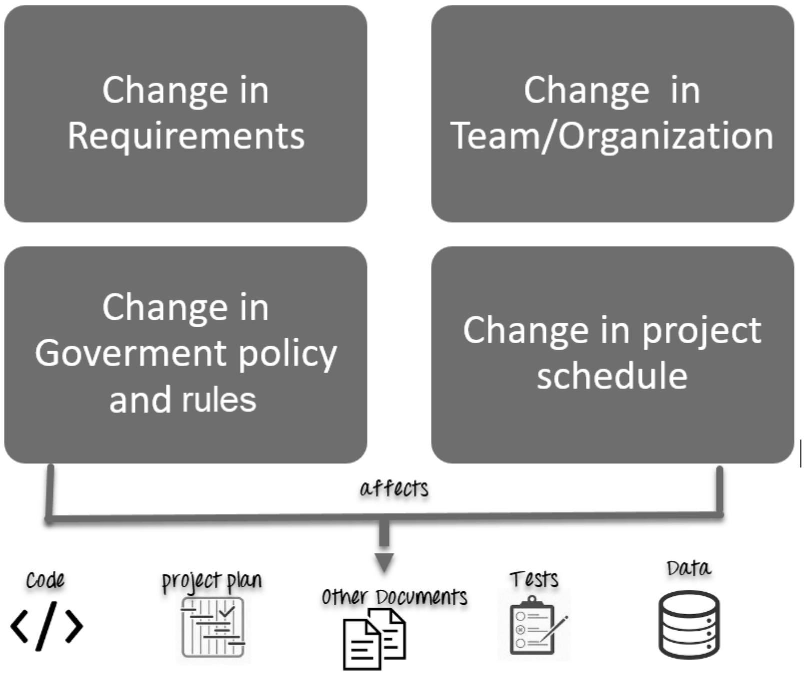
➢ *New customer needs* demand modification of data produced by information systems, functionality delivered by products, or services delivered by a computer-based system.

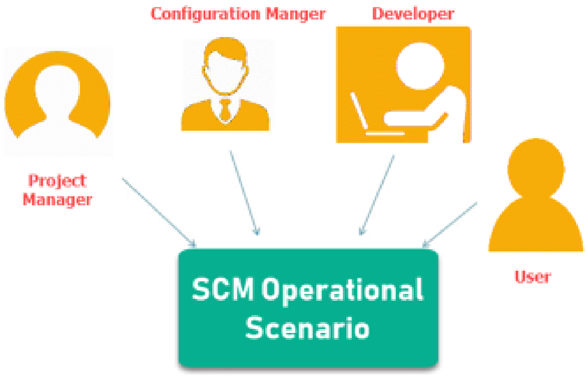
➢ *Reorganization or business growth/downsizing* causes changes in project priorities or software engineering team structure.

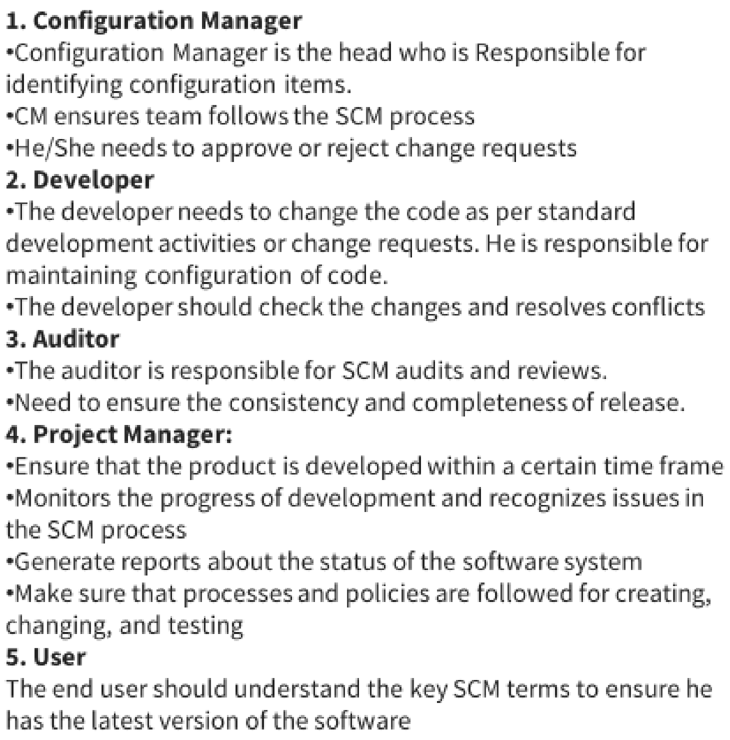
➢ *Budgetary or scheduling constraints* cause a redefinition of the system or product.

Software Configuration Management is a set of activities that have been developed to manage change throughout the life cycle of Computer Software.

- SCM can be viewed as a *Software Quality Assurance Activity* that is applied throughout the software process.


Change in Requirements
Change in Team/Organization
Change in Goverment policy and rules
Change in project schedule
affects
Code
project plan
Other Documents
Tests
Data




1. Configuration Manager
•Configuration Manager is the head who is Responsible for identifying configuration items.
•CM ensures team follows the SCM process
•He/She needs to approve or reject change requests
2. Developer
•The developer needs to change the code as per standard development activities or change requests. He is responsible for maintaining configuration of code.
•The developer should check the changes and resolves conflicts
3. Auditor
•The auditor is responsible for SCM audits and reviews.
•Need to ensure the consistency and completeness of release.
4. Project Manager:
•Ensure that the product is developed within a certain time frame
•Monitors the progress of development and recognizes issues in the SCM process
•Generate reports about the status of the software system
•Make sure that processes and policies are followed for creating, changing, and testing
5. User
The end user should understand the key SCM terms to ensure he has the latest version of the software

<u>Baselines</u>

- Change is a fact in software Development.
  - -Customers want to modify requirements
  - -Developers want to modify the technical approach
  - -Managers want to modify the project Strategy
- *A baseline is a software configuration management concept* that helps us to control change without seriously impeding justifiable change.
- *A baseline is a milestone* in the development of software that is marked by delivery of one or more software configuration items and the *approval of these SCIs* is obtained through a *Formal Technical Review.*
- ***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 control procedures.**
- *Before a software Configuration Items becomes a baseline, change may be made quickly and informally.*
- Changes can be made, but a specific, formal procedure must be applied to evaluate and verify each change.
- For example:

  Elements of a *Design Specification* have been documented and reviewed. Errors are found and corrected. Once all parts of the specification have been reviewed, corrected and then approved, the *Design Specification* becomes a baseline.

  Further changes to the program architecture (documented in the *Design Specification*) can be made only after each has been evaluated and approved.

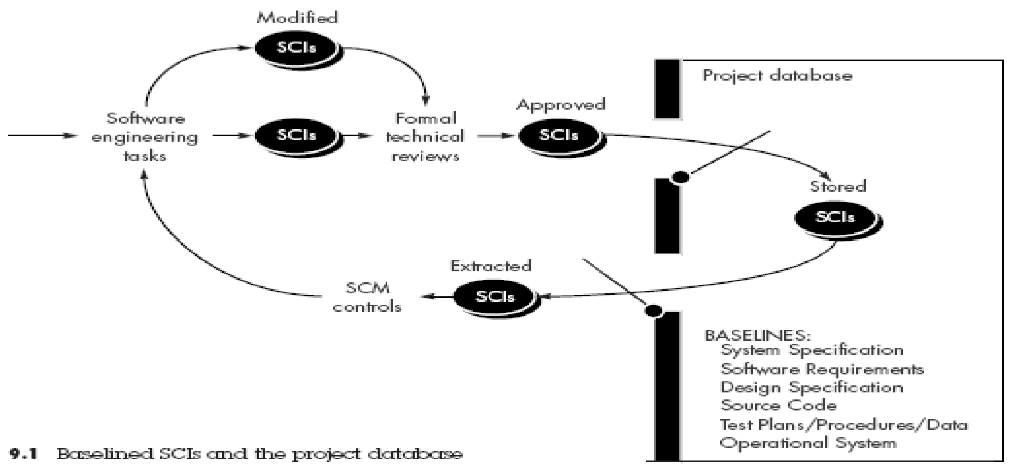


FIGURE 9.1 Baselined SCIs and the project database

The progression of events that lead to a baseline is also illustrated in Figure 9.1.

- Software engineering tasks produce one or more SCIs.
- After SCIs are reviewed and approved, they are placed in a *project database* (also called a *project library* or *software repository*).

- *When a member of a software engineering team wants to make a modification to a baselined SCI, it is copied from the project database into the engineer's private work space.*
- However, this extracted SCI can be modified only if SCM controls are followed.

➢ The arrows in Figure 9.1 illustrate the modification path for a baselined SCI.

<u>**Software Configuration Items**</u>

➢ Software Configuration Item is *information that is created/ generated as part of the software engineering process.*

➢ More realistically, *SCI is a document, an entire suite of test cases, or a named program component.*

➢ SCI could be considered to be a single section of a large specification or one test case in a large suite of tests.

➢ In addition to the SCIs that are derived from software work products, many software Engineering organizations also place *software tools* under configuration control.

➢ Because these tools were used to produce documentation, source code, and data, they must be available when changes to the software configuration are to be made.

➢ In reality, SCIs are organized to form *configuration objects* that may be cataloged in the project database with a single name.

➢ A configuration object has a *name, attributes, and is "connected" to other objects* by relationships.

➢ Referring to Figure 9.2 below, the configuration objects, **Design Specification, data model, component N, source code** and **Test Specification** are each defined separately.

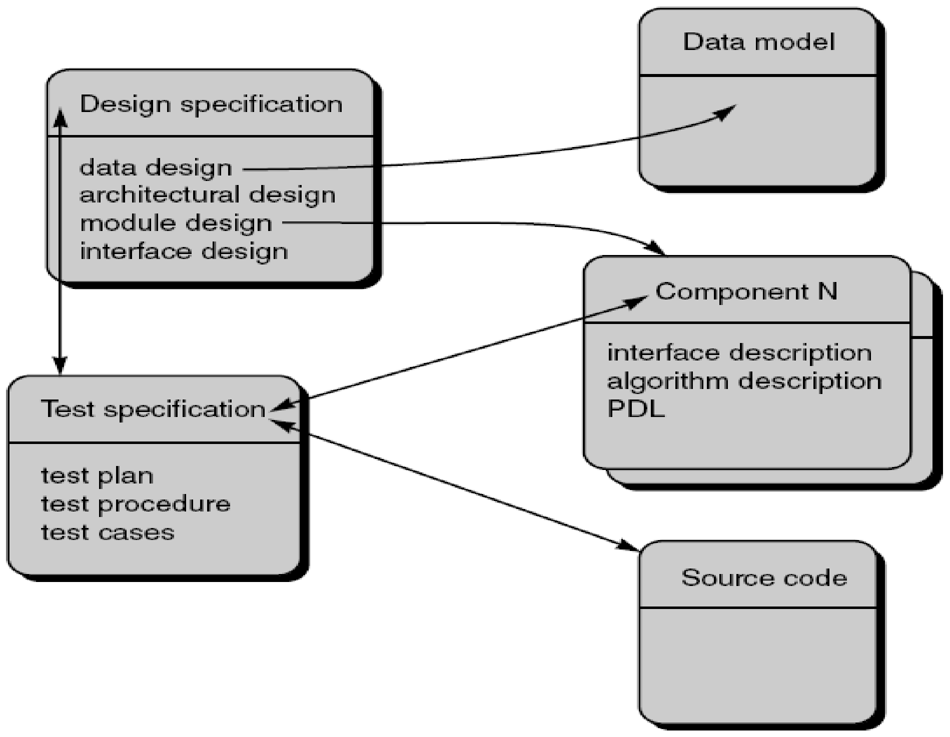
➢ However, each of the objects is related to the others as shown by the arrows. A curved arrow indicates a *compositional relation.*

➢ That is, **data model** and **component N** are part of the object **Design Specification.**

➢ A *double-headed straight arrow indicates an interrelationship.*

➢ If a change were made to the source code object, the interrelationships enable a software engineer to determine what other objects (and SCIs) might be affected.

FIGURE 9.2
Configuration objects



THE SCM Process

- ➢ Software configuration management is an *important element* of software quality assurance.
- ➢ Its primary responsibility is the *control of change.*
- ➢ However, SCM is also responsible for the *identification of individual SCIs* and various *versions of the software*, the *auditing* of the software configuration to ensure that it has been properly developed, and the *reporting* of all changes applied to the configuration.

➢ FIVE Software Configuration Management tasks

- *Identification* (tracking multiple versions to enable efficient changes)
- *Version control* (control changes before and after release to customer)
- *Change control* (authority to approve and prioritize changes)
- *Configuration auditing* (ensure changes made properly)
- *Reporting* (tell others about changes made)

IDENTIFICATION OF OBJECTS

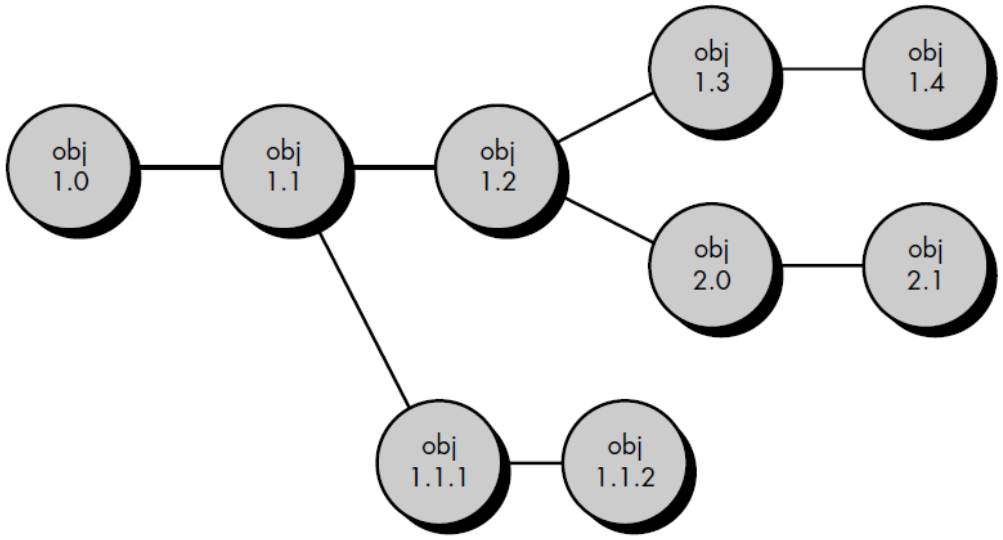
- To control and manage configuration items, each must be named and managed using an object-oriented approach.
- **Basic objects** are created by software engineers during analysis, design, coding, or testing.
- **Aggregate objects** are collections of basic objects and other aggregate objects. Design Specification is an aggregate object.
- Each objet has a set of distinct features (attributes):
  - unique name, description,
  - list of resources, realization
- Object Name : A character string that identifies the object unambiguously
- Object Description : a list of data items that identify
  - the SCI type (eg., document , program, data) represented by the object.
  - a project Identifier
  - change and/or version information
- Resources: Resources are the "*entities that are provided, processed, referenced or otherwise required by the object".* for example, data types, specific function or even variable name may be considered to be object resources.
- Realization : is the pointer to the "*unit of text*" to basic object and null for aggregate object.
- Configuration object identification must also consider the relationships that exist between named objects.
- An entity-relationship (E-R) diagram can be used to show the interrelationships among the objects.
- An object can be identified as a <*part of*> an aggregate object. The relationship <*part of*> defines a hierarchy of objects.
  Eg. ER diagram <*part of*> data model.
  data model <*part of*> specification.
- *The interrelationships between configuration objects can be represented with a* ***module interconnection language (MIL).*** A MIL describes the interdependencies among configuration objects and enables any version of a system to be constructed automatically.

Evolution Graph

- It is possible to create Evolution Graph for any object.
- The evolution graph describes the change history of an object, as illustrated in Figure 9.3

- Configuration object 1.0 undergoes revision and becomes object 1.1. Minor corrections and changes result in versions 1.1.1 and 1.1.2, which is followed by a major update that is object 1.2. The evolution of object 1.0 continues through 1.3 and 1.4, but at the same time, a major modification to the object results in a new evolutionary path, version 2.0.
- Both versions are currently supported.
- Changes may be made to any version, but not necessarily to all versions.

FIGURE 9.3
Evolution graph



<u>VERSION CONTROL</u>

- Version Control *Combines procedures and tools to manage the different versions of configuration objects created during the software process*
- *Each version* of the software is *collection of SCIs*(source, document, data) and each version may be composed of different variants.
- *Configuration management allows a user to specify alternative configuration of the software system through the selection of appropriate versions.*
- For example:

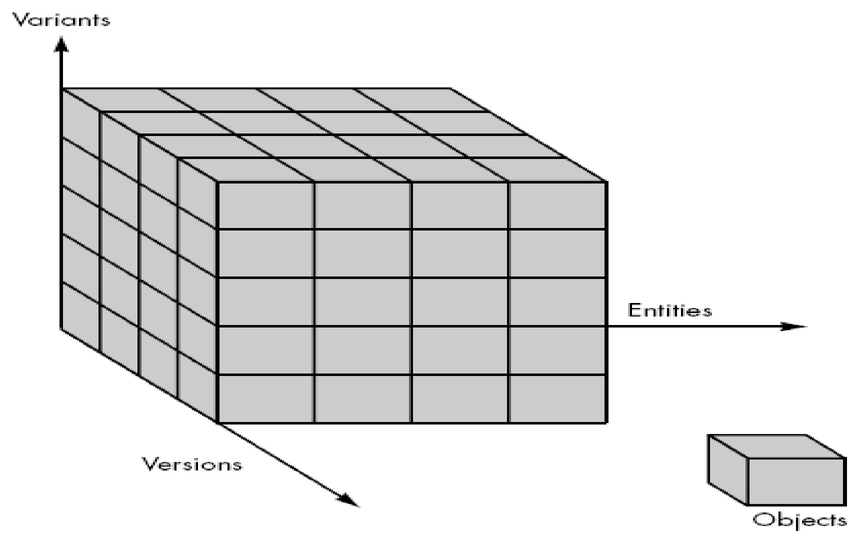
  A program composed of *entities 1,2,3,4 and 5*
  Entity 4 is only used when the software is implemented using *color Displays*.
  Entity 5 is only used when *Monochrome display* are available.

Therefore, two variants of the version can be defined:

1. entities 1,2,3 and 4 2. entities 1,2,3 and 5

- *Version Control* can be Represented using the evolution Graph or can be represented in a 3D space using an ***Object Pool***.
- Relationship between configuration objects and entities, variables and versions can be represented in a 3D space using a Object Pool concept.

FIGURE 9.4
Object pool representation of components, variants, and versions [REI89]



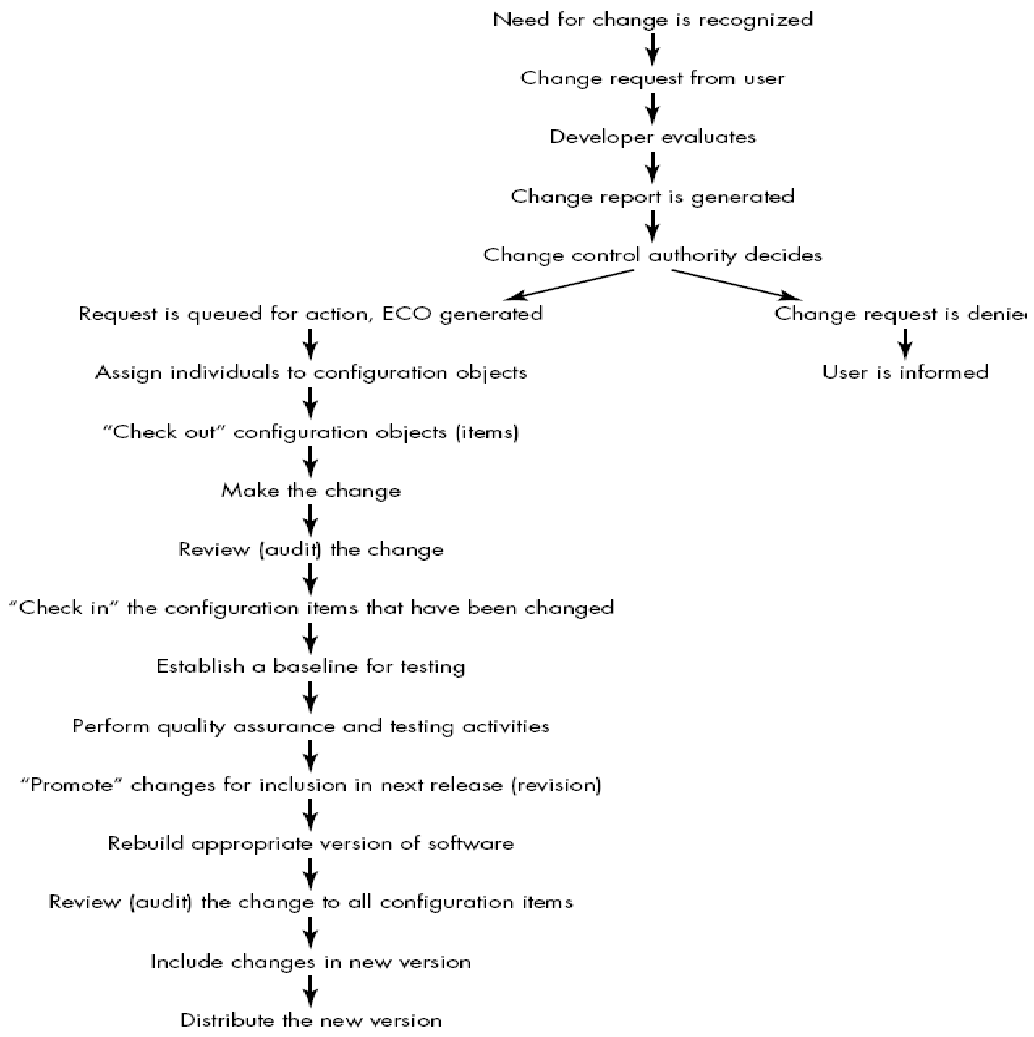
- An *entity is composed of collection of objects at the same revision level.*
- A *variant is a different set of objects at the same revision level* and coexists in parallel with other variants.
- A *new version* is defined when *major changes have been made* to one or more objects.
- A number of different automated approaches to version control have been purposed over past decades.
- A number of different automated approaches to version control have been proposed over the past decade. The primary difference in approaches is the sophistication of the attributes that are used to construct specific versions and variants of a system and the mechanics of the process for construction.

CHANGE CONTROL

- Change request is submitted and evaluated to assess technical merit and impact on the other configuration objects and budget.
- Change report contains the results of the evaluation.
- Change control authority (CCA) makes the final decision on the status and priority of the change based on the change report.
- Engineering change order (ECO) is generated for each change approved (describes change, lists the constraints, and criteria for review and audit)
- Object to be changed is checked-out of the project database subject to access control parameters for the object.
- Modified object is subjected to appropriate SQA and testing procedures.
- Modified object is checked-in to the project database and version control mechanisms are used to create the next version of the software.

- Synchronization control is used to ensure that parallel changes made by different people don’t overwrite one another.

FIGURE 9.5
The change control process


Need for change is recognized
Change request from user
Developer evaluates
Change report is generated
Change control authority decides
Request is queued for action, ECO generated
Change request is denie
User is informed
Assign individuals to configuration objects
"Check out" configuration objects (items)
Make the change
Review (audit) the change
"Check in" the configuration items that have been changed
Establish a baseline for testing
Perform quality assurance and testing activities
"Promote" changes for inclusion in next release (revision)
Rebuild appropriate version of software
Review (audit) the change to all configuration items
Include changes in new version
Distribute the new version

CONFIGURATION AUDIT

- Identification, version control, and change control help the software developer to maintain order in what would otherwise be a chaotic and fluid situation.
- To ***ensure that the change has been properly implemented*** we conduct :
  1) *formal technical reviews and*
  2) the software configuration audit.
- The **formal technical review** focuses on the technical correctness of the configuration object that has been modified.
- The reviewers assess the SCI to determine consistency with other SCIs, omissions, or potential side effects.
- A formal technical review should be conducted for all but the most trivial changes.

- A ***software configuration audit*** *complements the formal technical review by assessing a configuration object for characteristics that are generally not considered during review.*
- The audit asks and answers the following questions:

  1. Has the change specified in the ECO(Engineering change order) been made? Have any additional modifications been incorporated?

  2.Has a formal technical review been conducted to assess technical correctness?

  3. Has the software process been followed and have software engineering standards been properly applied?

  4. Has the change been "highlighted" in the SCI? Have the change date and change author been specified? Do the attributes of the configuration object reflect the change?

  5. Have SCM procedures for noting the change, recording it, and reporting it been followed?

  6. Have all related SCIs been properly updated?
- In some cases, the audit questions are asked as part of a formal technical review.
- However, when SCM is a formal activity, the SCM audit is conducted separately by the quality assurance group.

STATUS REPORTING

- *Configuration status reporting* (sometimes called *status accounting*) is an SCM task that answers the following questions:

  1) What happened?
  2) Who did it?
  3) When did it happen?
  4) What else will be affected?
- Each time an SCI is assigned new or updated identification, a CSR entry is made.
- Each time a change is approved by the CCA (i.e., an ECO is issued), a CSR entry is made.
- Each time a configuration audit is conducted, the results are reported as part of the CSR task.
- Configuration status reporting plays a vital role in the success of a large software development project.
- When many people are involved, it is likely that "the left hand not knowing what the right hand is doing" syndrome will occur.
- Two developers may attempt to modify the same SCI with different and conflicting intents.
- A software engineering team may spend months of effort building software to an obsolete hardware specification.
- The person who would recognize serious side effects for a proposed change is not aware

that the change is being made. CSR helps to eliminate these problems by improving communication among all people involved.

SCM Standards

- Over the past two decades a number of software configuration management standards have been proposed.
- Many early SCM standards, such as MIL-STD-483, DODSTD-480A and MIL-STD-1521A, focused on software developed for military applications.
- However, more recent ANSI/IEEE standards, such as ANSI/IEEE Stds. No. 828-1983, No. 1042-1987, and Std. No. 1028-1988, are applicable for nonmilitary software and are recommended for both large and small software engineering organizations.

Importance of version and change control

- Version control is used to combine procedures and tools to manage different versions of configurations objects that are made during software development.
- Change control adds human procedure with the automated tools so that effective control of change will be made.

Difference

Version Control	Change Control
Version control is to identify and manage project elements as they change over time.	Change control refers to the policy, rules, procedures, information, activities, roles, authorization levels and states relating to creation, updates, approvals, tracking and archiving of items involved with the implementation of a change process.