CHAPTER 5

SOFTWARE CONFIGURATION MANAGEMENT

In software engineering, software configuration management (SCM or S/W CM) is the task of tracking and controlling changes in the software, part of the larger cross-disciplinary field of configuration management. If something goes wrong, SCM can determine what was changed and who changed it. If a configuration is working well, SCM can determine how to replicate it across many hosts.

The goals of SCM are generally:

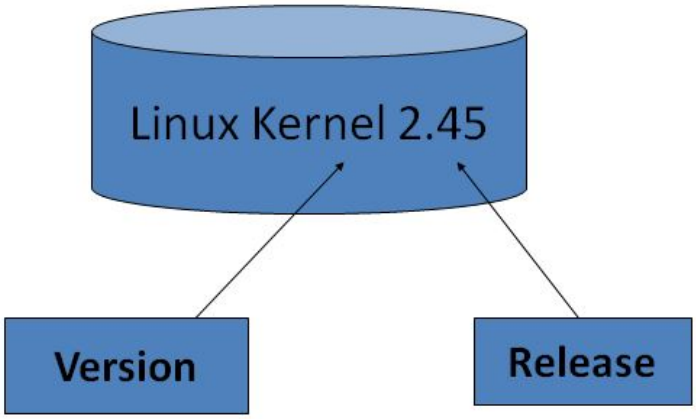
- Configuration identification - Identifying configurations, configuration items and baselines.
- Configuration control - Implementing a controlled change process. This is usually achieved by setting up a change control board whose primary function is to approve or reject all change requests that are sent against any baseline.
- Configuration status accounting - Recording and reporting all the necessary information on the status of the development process.
- Configuration auditing - Ensuring that configurations contain all their intended parts and are sound with respect to their specifying documents, including requirements, architectural specifications and user manuals.
- Build management - Managing the process and tools used for builds.
- Process management - Ensuring adherence to the organization's development process.
- Environment management - Managing the software and hardware that host the system.
- Teamwork - Facilitate team interactions related to the process.
- Defect tracking - Making sure every defect has traceability back to the source.

- New versions of software systems are created as they change:
  - For different machines/OS.
  - Offering different functionality.
  - Tailored for particular user requirements.
- Configuration management is concerned with managing evolving software systems:
  - System change is a team activity.
  - SCM aims to control the costs and effort involved in making changes to a system.
  - An Umbrella Activity.
  - For effectively tracking and controlling the configuration of software product.

- New Version:

  Significant change in functionality, technology, hardware and software requirements is called new version.

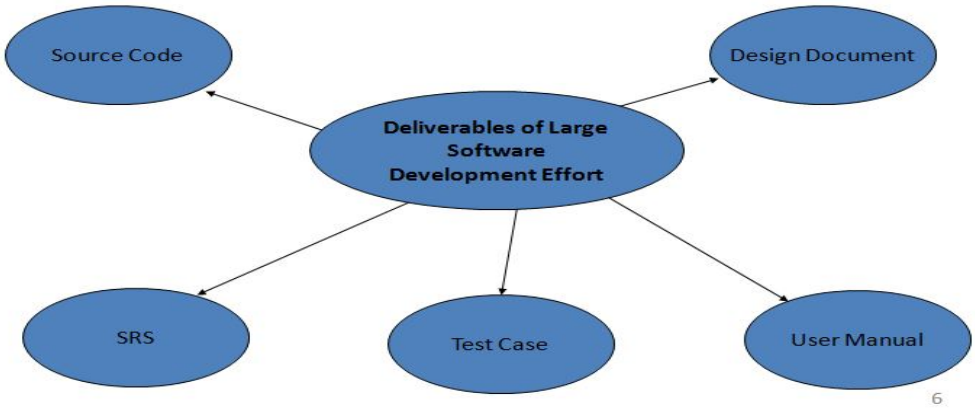
- New Release:
  Only a bug fix, minor enhancement in functionality is done when there is a new release.



In the above fig Linux Kernel 2 is the version number where as .45 is the release of that version.

❖ SOFTWARE CONFIGURATION ITEM:

All these items are referred to or managed by software engineer are known as software configuration item. Software configuration item (SCI) refers to the fundamental structural unit of a configuration management system. A release (itself a versioned entity) may consist of several configuration items. The set of changes to each configuration item will appear in the release notes, and the notes may contain specific headings for each configuration item.



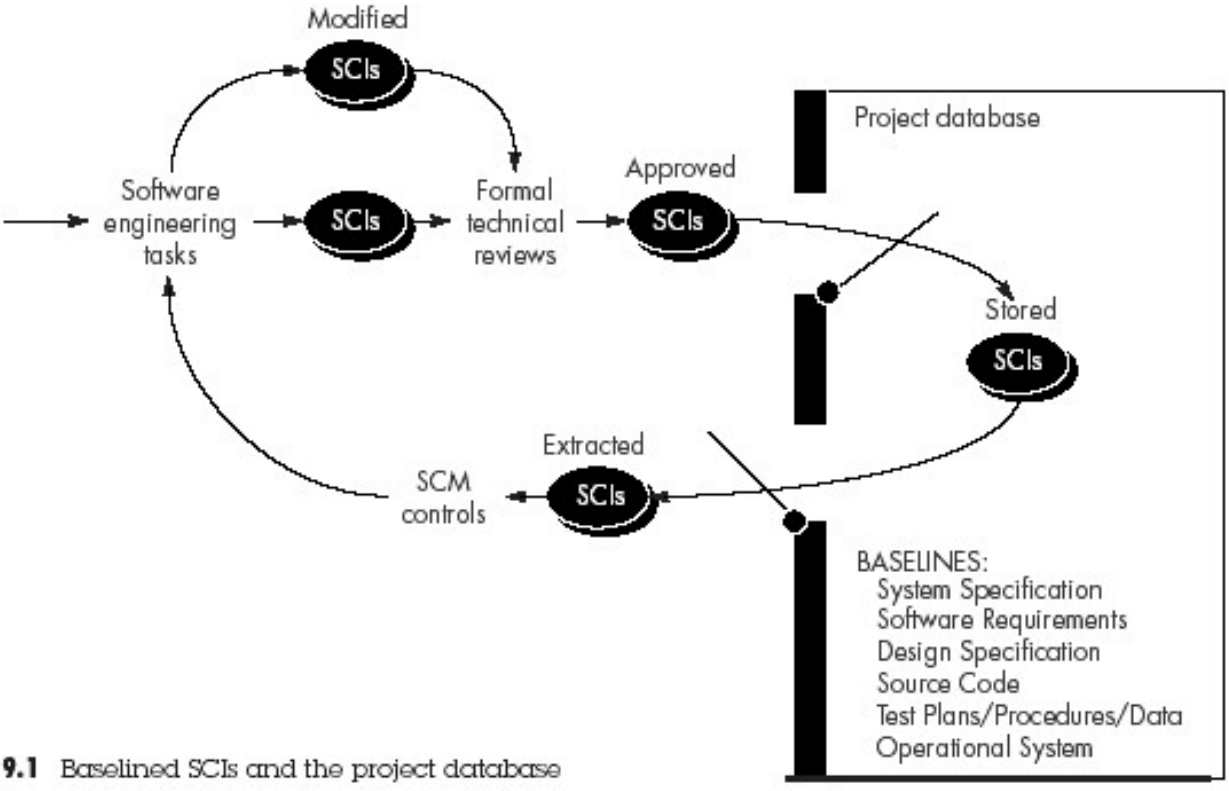
❖ A BASELINE:

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!! Before a software configuration item becomes a baseline, change may be made quickly and informally.

However, once a baseline is established, we figuratively pass through a swinging one way door. Changes can be made, but a specific, formal procedure must be applied to evaluate and verify each change.

How does SCI become Baseline?

- SCI obtain a private copy of module needing change.
- Perform all necessary changes.
- Get permission from the change control board for restoring changed module.
- After review from the CCB (Change Control Board) the manager updates the old baseline through restore operation.



9.1 Baselined SCIs and the project database

❖ SIGNIFICANCE OF SOFTWARE CONFIGURATION MANAGEMNET:

Because change can occur at any time, SCM activities are developed to

- Identify Change
- Control Change.
- Ensure that change is being properly implemented.
- Report changes to others who may have an interest.
- Control access to deliverables of software project.

❖ THE SCM PROCESS or SCM ACTIVITIES:

It is an important element for quality assurance. SCM is responsible for control of change. It also responsible for the identification of individual SCI (software configuration index) and various versions of the software, the auditing of the software configuration to ensure that it has been properly developed and reporting of all changes applied to the configuration.

Any discussion of SCM introduces a set of complex questions:

1. How does an organization identify and manage the existing version of program efficiently?
2. How does an organization control change before and after the s/w is released?
3. Who has the responsibility for approving ranking changes?
4. How to ensure the change has been made properly?
5. What mechanism is use to estimate order of change that are made?

These questions lead to the definition of five SCM tasks like:

- Identification
- Version control
- Change Control
- Auditing of Configuration
- Reporting

1. IDENTIFICATION OF PBJECTS IN SCM:

To control and manage s/w configuration item each must be separately named and then organized using object-oriented approach. Two types of objects can be identified i.e. basic objects and aggregate objects. Basic object is "unit of text" i.e requirement specifications, listing of components, suits of test cases to exercise code etc that has been created by the software engineer during analysis, design and test.

An aggregate object is collection of basic object. For eg: Design specificatio is aggregate object because it consists of basic objects like data model and other components. Each objects has a set of distinct features that identifies it uniquely like a name, a description, a list of resources and "realization". The object name is character string and description consists of list of data items. Configuration object identification must also consider the relationship that exists between the objects.

An object can be identified as <part-of> an aggregate objects. For eg:

data model<part-of>design specification

Change may be made to any version but not necessarily to all versions.

- How does the marketing department know customers currently have version 2?
- How can we be sure that change to version 2 source code is properly reflected in the documentation?

A key element in the answer to all these questions is identification.

2. VERSION CONTROL

- Version control is simply the automated act of tracking the changes of a particular file over time.
- This is typically accomplished by maintaining one copy of the file in a repository, then tracking the changes to that file (rather than maintaining multiple copies of the file itself).
- The concepts of check-in and check-out make this possible.
- Each time someone needs to edit a file, they check it out for exclusive editing and then check it back in to the repository when finished, thus creating a new version. This paradigm of check-in and check-out can be as simple as I describe above or much more

complex depending on the amount of parallel development and branching you have in your process.

Benefits of version control:

1. Roll back to a previous version of a given file
2. Compare two versions of a file, highlighting differences
3. Provide a mechanism of locking, forcing serialized change to any given file
4. Create branches that allow for parallel concurrent development
5. Maintain an instant audit trail on each and every file: versions, modified date, modifier, and any additional amount of metadata your system provides for and you choose to implement.

3. CHANGE CONTROL :

- Change control combines human procedures and automated tools to provide a mechanism for the control of change.

- The change control process is usually conducted as a sequence of steps proceeding from the submission of a change request. Typical IT change requests include the addition of features to software applications, the installation of patches, and upgrades to network equipment.

Here's an example of a six-step process for a software change request:

a. **Documenting the change request:**
When the client requests the change, that request is categorized and recorded, along with informal assessments of the importance of that change and the difficulty of implementing it.

b. **Formal assessment:**
The justification for the change and risks and benefits of making/not making the change are evaluated. If the change request is accepted, a development team will be assigned. If the change request is rejected, that fact is documented and communicated to the client.

c. **Planning:**

The team responsible for the change creates a detailed plan for its design and implementation, as well as a plan for rolling back the change should it be deemed unsuccessful.

d. **Designing and testing:**

The team designs the program for the software change and tests it. If the change is deemed successful, the team requests approval and a date for implementation.

e. **Implementation and review:**

The team implements the program and stakeholders review the change.

f. **Final assessment:**

If the client is satisfied that the change was implemented satisfactorily, the change request is closed. If the client is not satisfied, the project is reassessed and steps may be repeated.

4. CONFIGURATION ADUIT:

- The FTR focuses on the technical correctness of the configuration objects that have been modified.
- Configuration audit complements the FTR by assessing configuring objects that are not considered during review.
- **Configuration Audit** (CA) is the formal examination of the "as-built" configuration of a configuration item against its technical documentation to establish or verify the configuration item's product baseline.
- The CA is used to examine the actual configuration of the Configuration Item (CI) that is representative of the product configuration in order to verify that the related design documentation matches the design of the deliverable CI.
- It is also used to validate many of the supporting processes that the contractor uses in the production of the CI.
- The audit asks and answers the following questions:

1. Has the specified change been made? have any additional modifications been incorporated?
2. Has FTR been conducted to assess technical correctness?

3. Have all related SCIs been properly updated?

5. STATUS REPORTING

It is sometimes called status accounting. Status reporting mostly provides answers to these questions:

1. What happened?

2. Who did it?

3. When did it happen?

4. What else will be affected?

- Each time change is assigned new or update identification, a CSR (Configuration status reporting) entry is made.
- Output of CSR may be placed in database so that s/w developers and maintainers can access changed information by keyword categories.
- CSR is generated on a regular basis and plays a vital role in the success of the large s/w development projects.
- CSR helps to eliminate the problems like not being aware of changes and its side effects by improving communication among all people involved.

❖ SRS: SOFTWARE REQUIEMENT SPECIFICATION

SRS in context of Software Engineering stands for System Requirements Specification. It is a document that specifies the complete description of the behavior of the system.

For example, if the group of software engineers are to design a software for a bank. Assuming they are provided with BRD (Business Requirement Design), the engineers first need to describe and design the behavior of the software. The various entities the software has to react with, the various properties it should possess and so on. These specification can be a type of SRS.

General structure of SRS:

1. Introduction

- Purposes
- Scope
- Definitions
- System Overview
- References

2. Overall Description
    - Product Perspective Product Functions
    - User Characteristics
    - Constraints, assumptions, and Dependencies
3. Specific Requirements
    - External interfaces
    - Functions
    - Performance requirements
    - Logical database requirement
    - Design constraints
    - Key features

❖ SCM STANDARDS:

This standard establishes the minimum required contents of a Software Configuration Management (SCM) Plan is called SCM standard. This standard applies to the entire life cycle of critical software; e.g., where failure would impact safety or cause large financial or social losses. It also applies to noncritical software and to software already developed. The application of this standard is not restricted to any form, class, or type of software.

The standard that is used in software configuration management is 828-2012 - IEEE Standard for Configuration Management in Systems and Software Engineering. This standard establishes the minimum requirements for processes for Configuration Management (CM) in systems and software engineering. The application of this standard applies to any form, class, or type of software or system.

This revision of the standard expands the previous version to explain CM, including identifying and acquiring configuration items, controlling changes, reporting the status of configuration items, as well as software builds and release engineering. Its predecessor defined only the contents of a software configuration management plan. This standard addresses what CM activities are to be done, when they are to happen in the life cycle, and what planning and resources are required. It also describes the content areas for a CM Plan.