Software Configuration Management

When we develop software, the product (software) undergoes many changes in their maintenance phase; we need to handle these changes effectively.

Several individuals (programs) works together to achieve these common goals. This individual produces several work product (SC Items) e.g., Intermediate version of modules or test data used during debugging, parts of the final product.

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

As software development progresses, the number of Software Configuration elements (SCI's) grow rapidly.

These are handled and controlled by SCM. This is where we require software configuration management.

A configuration of the product refers not only to the product's constituent but also to a particular version of the component.

Therefore, SCM is the discipline which

- Identify change
- Monitor and control change
- Ensure the proper implementation of change made to the item.
- Auditing and reporting on the change made.

Configuration Management (CM) is a technic of identifying, organizing, and controlling modification to software being built by a programming team.

The objective is to maximize productivity by minimizing mistakes (errors).

CM is used to essential due to the inventory management, library management, and updation management of the items essential for the project.

Why do we need Configuration Management?

Multiple people are working on software which is consistently updating. It may be a method where multiple version, branches, authors are involved in a software project, and the team is geographically distributed and works concurrently. It changes in user requirements, and policy, budget, schedules need to be accommodated.

Importance of SCM

It is practical in controlling and managing the access to various SCIs e.g., by preventing the two members of a team for checking out the same component for modification at the same time.

It provides the tool to ensure that changes are being properly implemented.

It has the capability of describing and storing the various constituent of software.

SCM is used in keeping a system in a consistent state by automatically producing derived version upon modification of the same component.

SCM Process

It uses the tools which keep that the necessary change has been implemented adequately to the appropriate component. The SCM process defines a number of tasks:

- Identification of objects in the software configuration
- Version Control
- Change Control
- Configuration Audit
- Status Reporting

Identification

Basic Object: Unit of Text created by a software engineer during analysis, design, code, or test.

Aggregate Object: A collection of essential objects and other aggregate objects. Design Specification is an aggregate object.

Each object has a set of distinct characteristics that identify it uniquely: a name, a description, a list of resources, and a "realization."

The interrelationships between configuration objects can be described with a Module Interconnection Language (MIL).

Version Control

Version Control combines procedures and tools to handle different version of configuration objects that are generated during the software process.

Clemm defines version control in the context of SCM: Configuration management allows a user to specify the alternative configuration of the software system through the selection of appropriate versions. This is supported by associating attributes with each software version, and then allowing a configuration to be specified [and constructed] by describing the set of desired attributes.

Change Control

James Bach describes change control in the context of SCM is: Change Control is Vital. But the forces that make it essential also make it annoying.

We worry about change because a small confusion in the code can create a big failure in the product. But it can also fix a significant failure or enable incredible new capabilities.

We worry about change because a single rogue developer could sink the project, yet brilliant ideas originate in the mind of those rogues, and

A burdensome change control process could effectively discourage them from doing creative work.

A change request is submitted and calculated to assess technical merit; potential side effects, the overall impact on other configuration objects and system functions, and projected cost of the change.

The results of the evaluations are presented as a change report, which is used by a change control authority (CCA) - a person or a group who makes a final decision on the status and priority of the change.

The "check-in" and "check-out" process implements two necessary elements of change control- **access control** and **synchronization control**.

Access Control governs which software engineers have the authority to access and modify a particular configuration object.

Synchronization Control helps to ensure that parallel changes, performed by two different people, don't overwrite one another.

Configuration Audit

SCM audits to verify that the software product satisfies the baselines requirements and ensures that what is built and what is delivered.

SCM audits also ensure that traceability is maintained between all CIs and that all work requests are associated with one or more CI modification.

SCM audits are the "**watchdogs**" that ensures that the integrity of the project's scope is preserved.

Status Reporting

Configuration Status reporting (sometimes also called status accounting) providing accurate status and current configuration data to developers, testers, end users, customers and stakeholders through admin guides, user guides, FAQs, Release Notes, Installation Guide, Configuration Guide, etc.

Software Maintenance Cost Factors

There are two types of cost factors involved in software maintenance.

These are

- Non-Technical Factors
- Technical Factors

Non-Technical Factors

1. Application Domain

- If the application of the program is defined and well understood, the system requirements may be definitive and maintenance due to changing needs minimized.
- If the form is entirely new, it is likely that the initial conditions will be modified frequently, as user gain experience with the system.

2. Staff Stability

- It is simple for the original writer of a program to understand and change an application rather than some other person who must understand the program by the study of the reports and code listing.
- If the implementation of a system also maintains that systems, maintenance costs will reduce.
- In practice, the feature of the programming profession is such that persons change jobs regularly. It is unusual for one user to develop and maintain an application throughout its useful life.

3. Program Lifetime

- Programs become obsolete when the program becomes obsolete, or their original hardware is replaced, and conversion costs exceed rewriting costs.

4. Dependence on External Environment

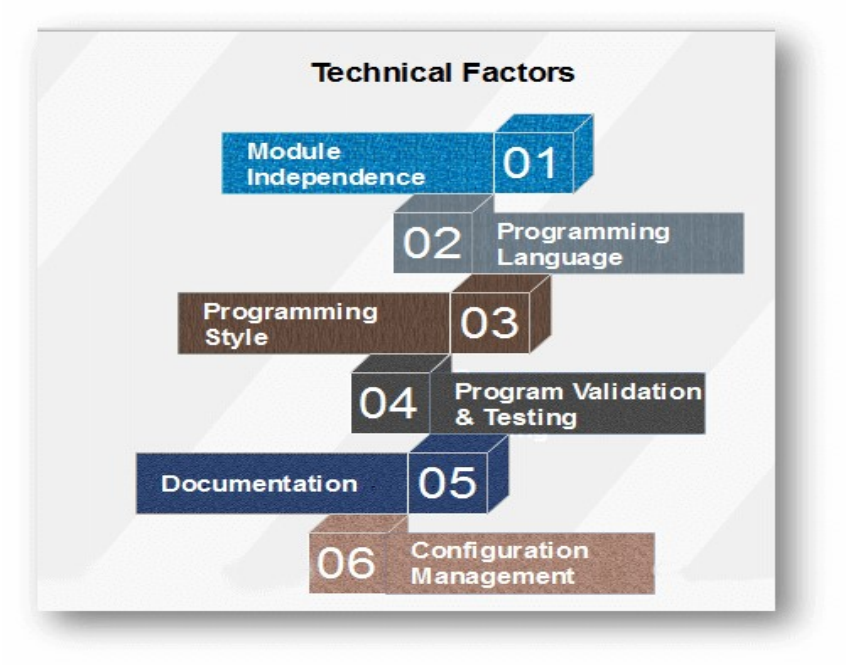
- If an application is dependent on its external environment, it must be modified as the climate changes.
- For example:
- Changes in a taxation system might need payroll, accounting, and stock control programs to be modified.
- Taxation changes are nearly frequent, and maintenance costs for these programs are associated with the frequency of these changes.
- A program used in mathematical applications does not typically depend on humans changing the assumptions on which the program is based.

5. Hardware Stability

- If an application is designed to operate on a specific hardware configuration and that configuration does not changes during the program's lifetime, no maintenance costs due to hardware changes will be incurred.
- Hardware developments are so increased that this situation is rare.
- The application must be changed to use new hardware that replaces obsolete equipment.

Technical Factors

Technical Factors include the following:



Module Independence

It should be possible to change one program unit of a system without affecting any other unit.

Programming Language

Programs written in a high-level programming language are generally easier to understand than programs written in a low-level language.

Programming Style

The method in which a program is written contributes to its understandability and hence, the ease with which it can be modified.

Program Validation and Testing

- Generally, more the time and effort are spent on design validation and program testing, the fewer bugs in the program and, consequently, maintenance costs resulting from bugs correction are lower.
- Maintenance costs due to bug's correction are governed by the type of fault to be repaired.
- Coding errors are generally relatively cheap to correct, design errors are more expensive as they may include the rewriting of one or more program units.

- Bugs in the software requirements are usually the most expensive to correct because of the drastic design which is generally involved.

Documentation

- If a program is supported by clear, complete yet concise documentation, the functions of understanding the application can be associatively straight-forward.
- Program maintenance costs tends to be less for well-reported systems than for the system supplied with inadequate or incomplete documentation.

Configuration Management Techniques

- One of the essential costs of maintenance is keeping track of all system documents and ensuring that these are kept consistent.
- Effective configuration management can help control these costs.