Unit-7

Software Configuration Management

Concept

Software Configuration Management (SCM)

- The art of identifying, organizing, and controlling modifications to the software being built.
- The main goal is to minimize mistakes.
- It is an umbrella activity, applied throughout the software engineering process.
- Auditing of the software configuration
- Reporting of all the changes applied to configuration
- Control of change

Requirement and Elements of SCM

1. Identification, control, audit, and status accounting are the four basic requirements for a software configuration management system.
2. Identification:
   a) Each software part is labeled so that it can be identified.
   b) Furthermore, there will be different versions of the software parts as they evolve over time, so a version or revision number will be associated with the part.
   c) The key is to be able to identify any and all artifacts that compose a released configuration item.
   d) It helps to answer the following question.
      i. What is the component of the product?
      ii. What is the version of configured item?
   e) Think of this as a bill of materials for all the components in your automobile. When the manufacturer realizes that there has been a problem with parking brakes purchased from a subcontractor, it needs to know all the automobile models using that version of the parking brake.

Baseline

In software configuration management (SCM), a baseline is a formally approved and documented version of a software product at a specific point in time. Think of it as a snapshot that serves as a reference point for tracking changes and ensuring project stability.

A baseline is a formally accepted version of a software configuration item. It is designated and fixed at a specific time while conducting the SCM process. It can only be changed through formal change control procedures.

Activities during this process:

1. Facilitate construction of various versions of an application
2. Defining and determining mechanisms for managing various versions of these work products
3. The functional baseline corresponds to the reviewed system requirements
4. Widely used baselines include functional, developmental, and product baselines

In simple words, baseline means ready for release.

A baseline is a milestone in the development of software that marked the delivery of one or more software configuration items.

IEEE define baseline as: "A specification or product that has been formally reviewed and agreed upon, that thereafter serve as the basis for further development, and that can be changed only through formal change control procedure"

Common Baselines:

1. Software Engineering /Specification
2. Requirement Analysis
3. Software design
4. Coding
5. Testing
6. Release

Benefits of using baselines:

- **Reduced errors:** Tracking changes against a baseline helps identify and prevent regressions or unintended modifications.
- **Improved visibility:** Everyone involved in the project can clearly understand the current state of the software and any planned changes.
- **Streamlined change management:** Formalized change control processes based on baselines help ensure controlled and documented modifications.
- **Simplified quality assurance:** Testing efforts can focus on changes made since the last baseline to ensure quality is maintained.

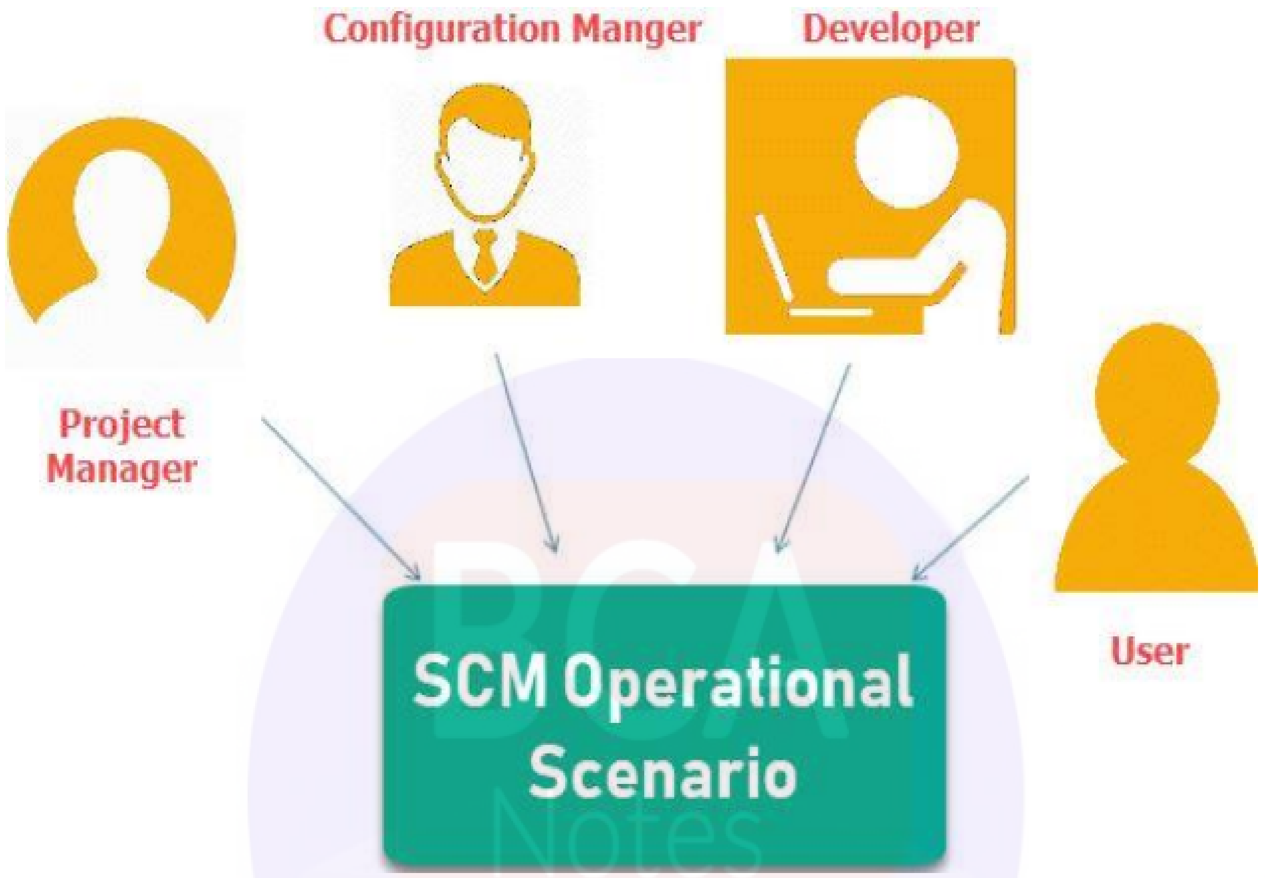
SCM Repository

An SCM repository is a place where developers store and manage all the files and data related to a software project. It acts as a central location for storing source code, documentation, configuration files, and other project assets. The repository keeps track of different versions of files, allowing developers to access, modify, and share project resources collaboratively. Additionally, it provides version control features, enabling teams to track changes, revert to previous versions if needed, and ensure consistency and integrity throughout the development process. Essentially, the SCM repository serves as a secure and organized hub for managing and controlling the evolution of software projects.

The repository performs or precipitates the following functions:

1. Data integrity
2. Information sharing
3. Tool integration
4. Data integration
5. Methodology enforcement
6. Document standardization

Participation of SCM Repository



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

Versioning & Version Control

Versioning:

- **Definition:** The act of assigning unique identifiers to different versions of a software product or artifact. These identifiers can be simple numbers (v1.0, v2.1), letters (A, B, C), or more complex schemes.
- **Purpose:** Allows for clear distinction between different versions, enabling users to identify the specific version they are using and track changes over time.
- **Examples:** Software releases, documentation revisions, design prototypes.

Version Control:

- **Definition:** A system that tracks and manages changes made to software files and documents. It stores different versions of files, allows users to revert to previous versions, and helps collaborate on projects by preventing conflicts.
- **Purpose:** Enables developers to work on the same codebase without overwriting each other's work, facilitates collaboration and code reviews, and provides a historical record of changes.
- **Examples:** Git, SVN, Mercurial, CVS.

Benefits of Versioning & Version Control:

- **Improved traceability:** Track the evolution of code and identify who made what changes.
- **Easier collaboration:** Multiple developers can work on the same codebase without conflicts.
- **Rollback capability:** Allows reverting to previous versions if problems arise.
- **Enhanced quality assurance:** Enables testing of specific versions.
- **Improved documentation:** Keeps track of changes made to documents and code.

SCM Process

The SCM process, or Software Configuration Management process, is a set of activities and tools used to control and track changes made to software throughout its lifecycle. It ensures that the software remains stable, consistent, and meets its intended purpose**.**

Here's a breakdown of the key stages involved:

1. Identification:

- Identify all elements that need to be controlled, including source code, documentation, test cases, configurations, etc.
- Define clear naming conventions and versioning schemes for each element.

2. Version control:

- Implement a version control system like Git or SVN to store different versions of each element and track changes made.
- Enforce controlled access and change management procedures to prevent conflicts and maintain traceability.

3. Change management:

- Establish a formal process for proposing, reviewing, and approving changes to controlled elements.
- This process ensures changes are necessary, well-defined, and don't negatively impact other parts of the software.

4. Configuration management:

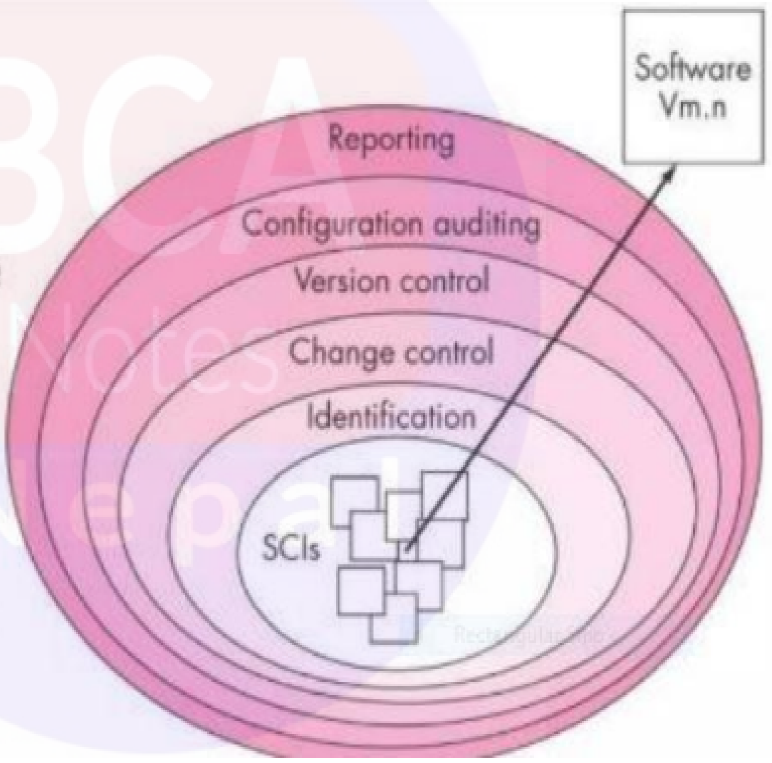
- Define and manage different configurations of the software for different environments (development, testing, production, etc.).
- Ensure consistency and compatibility between different configurations.

5. Status reporting and auditing:

- Regularly track the status of changes and configurations to maintain visibility and control.
- Conduct audits to verify compliance with established procedures and identify potential issues.

Primary Objectives:

1. To identify all items that collectively define the software configuration
2. To manage changes to one or more of these items
3. To facilitate the construction of different versions of an application
4. To ensure that software quality is maintained as the configuration evolves over time



Change Control Process

The change control process (CCP) is a formalized, structured approach used to manage and approve changes made to any product, project, or service, including software. It ensures that proposed changes are evaluated for impact, risks, and potential benefits before being implemented.

Different Types of Change Requests throughout the life of your project, there will be two kinds of changes that occur:

1. Changes beyond your control: weather events, team attrition, or employee sickness.
2. Changes that are specifically requested: Adding features to software applications, updating network equipment, Installing system patches

Key Steps:

1. **Initiation:** A change request is proposed, outlining the desired change, rationale, and potential impact.
2. **Evaluation:** The request is reviewed by a designated team to assess its feasibility, risks, and benefits. This might involve technical analysis, cost estimation, and impact on other parts of the system.
3. **Approval:** Based on the evaluation, the request is approved, rejected, or modified. Decisions are documented and communicated to all stakeholders.
4. **Implementation:** If approved, the change is implemented according to a defined plan, which may involve testing, documentation updates, and training.
5. **Review and Closure:** The effectiveness of the implemented change is reviewed, lessons learned are documented, and the change is formally closed.

Benefits:

- **Reduced risks:** CCP helps identify and mitigate potential negative consequences of changes.
- **Improved quality:** Ensures changes are well-defined, tested, and meet quality standards.
- **Increased transparency:** Keeps stakeholders informed of proposed and implemented changes.
- **Better cost control:** Allows for careful evaluation of cost implications before approval.
- **Reduced rework:** Minimizes the need to undo poorly planned or implemented changes.

Change Control Procedures in Software Project Management

Step 1: Document the Change Request

When a change request occurs, your first step is to categorize and record it. This is required even if you ultimately decide not to pursue it.

Step 2: Conduct a Formal Change Evaluation

Your project team will meet and formally evaluate the change. Key questions to ask include:

- Is the change justified?
- What are the risks and benefits of making the change?
- What are the risks of not making the change?
- Do the risks outweigh the benefits?
- If you decide to accept a change, you'll assign the next steps to a dedicated development team.

Step 3: Plan the Change

- The team that you task with developing the change will create a detailed plan outlining how it will be designed and implemented.
- At this juncture, there should be methods in place to define and verify success.

Step 4: Design Software Changes

If the requested change affects your ERP software, the team will design the new program and test it for accuracy. If it passes successfully, they'll request approval and set a date for implementation.

Step 5: Conduct an Internal Software Review

If the software change is approved, the team will proceed to implement the change. Then, they'll perform demonstrations for stakeholders so they can review and accept the final change.

Step 6: Conduct a Final Assessment

If everything is implemented and reviewed successfully, the status of the change request will change from active to closed.

Configuration Audit & Status Reporting

Configuration Audit

Focus: Examining and verifying the current state of a software product against its documented configuration.

Activities:

- Identifying discrepancies or unauthorized changes.
- Assessing the impact of any deviations.
- Verifying consistency across environments.
- Employing methods like physical/functional/documentation audits.

Outcome: Raw data and findings about the configuration's adherence to documentation.

Configuration Status Reporting

Focus: Presenting the findings of the audit in a clear and concise format.

Content:

- Summary of audit findings with impact analysis.
- Recommendations for corrective actions.
- Status of change requests and their potential impact.
- Metrics on configuration stability and change control effectiveness.

- **Outcome**: Distilled information and recommendations for stakeholders to make informed decisions.

Key Differences:

- **Purpose:** Audit finds issues, reporting communicates them.
- **Audience:** Audit for internal teams, reporting for broader stakeholders.
- **Action:** Audit gathers data, reporting recommends solutions.
- **Output:** Audit findings as raw data, reporting as processed information.

Case Study: Version Control Software Tools

(Git, CVS, SVN)

This case study compares three commonly used version control software tools: Git, CVS, and SVN. Each tool offers unique features and benefits, suitable for different project needs and workflows.

1. Git:

- **Distributed system:** Each developer has a complete copy of the repository, enabling offline work and easier branching.
- **Strong focus on branching and merging:** Enables simultaneous work on features without affecting the main codebase.
- **Command-line interface:** Can be powerful for experienced users, but requires learning curve for beginners.
- **Popular for large, complex projects with active collaboration.**

2. CVS (Concurrent Versions System):

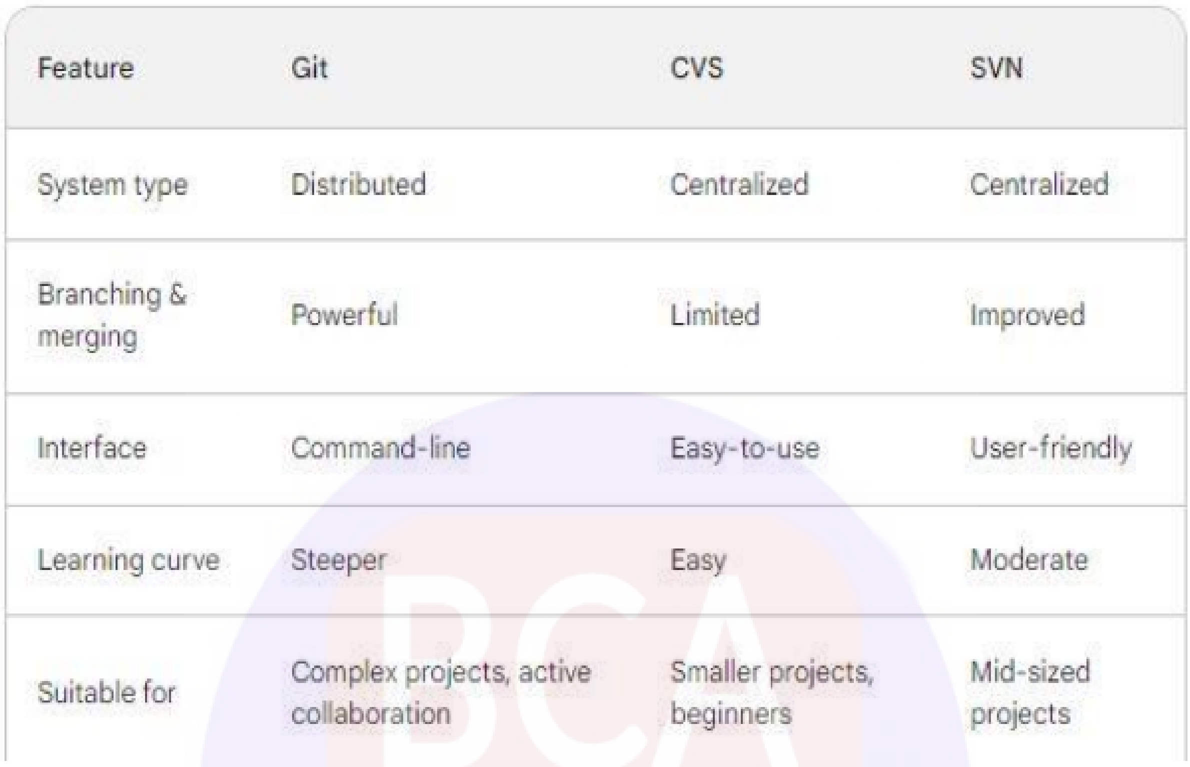
- Centralized system: A single server stores the repository, requiring constant network access.
- Simple branching and merging: Limited branching capabilities compared to Git.
- Easy to learn and use: User-friendly interface makes it suitable for beginners.

- Better suited for smaller projects or teams with less complex branching needs.

3. SVN (Apache Subversion):

- Centralized system similar to CVS but more robust and scalable.
- Improved branching and merging: Offer better capabilities than CVS for managing branches.
- User-friendly interface with command-line and GUI options.
- A good balance between ease of use and features for mid-sized projects.

Comparison Table:



Feature	Git	CVS	SVN
System type	Distributed	Centralized	Centralized
Branching & merging	Powerful	Limited	Improved
Interface	Command-line	Easy-to-use	User-friendly
Learning curve	Steeper	Easy	Moderate
Suitable for	Complex projects, active collaboration	Smaller projects, beginners	Mid-sized projects