SOFTWARE CONFIGURATION MANAGEMENT (SCM)


Learning Objectives

- To become familiar with the concepts of software configuration management along with its key elements
- To understand the main tasks and responsibilities of software configuration management
- To be able to develop a configuration management plan
First Law of Systems Engineering

- No matter where you are in the system life cycle, the system will change, and the desire to change it will persist throughout the lifecycle.
- Change is the only constant
- Why?

Configuration Management

**Definition:**
The set of activities that have been developed to manage change throughout the software life cycle.

**Purpose:**
Systematically control changes to, and maintain the integrity and traceability of the configuration throughout the system’s life cycle.
Baseline

Definition: Specification or product that
- has been formally reviewed and agreed upon,
- serves as the basis for further development, and
- can be changed only through formal change control procedures.

One “official version” at any point in time

Might signal a point of departure from one activity to the start of another activity.

Helps control change without impeding justifiable change.

Baseline (Cont.)

Central repository of reviewed and approved artifacts that represent a given stable point in overall system development.

Shared DB for project and kept in consistent state.

Policies allow the team to achieve consistent state and manage the project.
Software Configuration Item (SCI)

- Definition: Information that is created as part of the software engineering process.

- Examples:
  - Software Project Plan
  - Software Requirements Specification
    - Models, Prototypes, Requirements
  - Design document
  - Source code
  - Test suite
  - Software tools (e.g., compilers) Why?

Elements of SCM

There are four elements of SCM:
1. Software Configuration Identification
2. Software Configuration Control
3. Software Configuration Auditing
4. Software Configuration Status Accounting
Software Configuration Identification

- Provides labels for the baselines and their updates.
- Evolution graph: depicts versions/variants.

Software Configuration Control

Three basic ingredients to SCC

1. **Documentation** for formally precipitating and defining a proposed change to a software system.
2. **An organizational body** (Configuration Control Board) for formally evaluating and approving or disapproving a proposed change to a software system.
3. **Procedures** for controlling changes to a software system.
Software Configuration Control

- Why needed?
  - Not all possible changes are beneficial.
  - Need a mechanism to control access to different items of the configuration (who can access what and when).

Access and Synchronization Control

Check-In

Check-Out

Software Engineer

Access Control

Project Database

Configuration Object (Baseline Version)

Configuration Object (Modified Version)

Audit Info

Unlock

Ownership Info

Lock

Configuration Object Request
Access and Synchronization Control

**Check-In**
- Software Engineer
- Access Control
- Check-Out
- Configuration Object (Baseline Version)
- Ownership Info
- Audit Info
- Unlock
- Configuration Object (Baseline Version)
- Project Database
- Configuration Object (Extracted Version)
- Configuration Object (Modified Version)

**Check-Out**
- Software Engineer
- Access Control
- Check-In
- Configuration Object (Baseline Version)
- Ownership Info
- Audit Info
- Unlock
- Configuration Object (Baseline Version)
- Project Database
- Configuration Object (Extracted Version)
- Configuration Object (Modified Version)
Access and Synchronization Control

Configuration Management Cycle

<table>
<thead>
<tr>
<th>Role</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Customer generates a change request</td>
</tr>
<tr>
<td></td>
<td>Customer approves changes</td>
</tr>
<tr>
<td>Project Manager</td>
<td>Manager assigns change request to software engineer</td>
</tr>
<tr>
<td>Dev Team</td>
<td>Software engineer checks out necessary configuration objects</td>
</tr>
<tr>
<td></td>
<td>SE completes necessary changes</td>
</tr>
<tr>
<td></td>
<td>SE checks in modified configuration objects and notifies CM</td>
</tr>
<tr>
<td>Configuration Manager*</td>
<td>CM creates new system baseline</td>
</tr>
<tr>
<td></td>
<td>CM prepares new system release for operation if necessary</td>
</tr>
</tbody>
</table>

*In charge of administering project database and providing access control to engineers
Software Configuration Auditing

Allow to follow trails, e.g.,
- Which changes have been made
- Who did those changes, and
- Why

Provides mechanism for determining the degree to which the current configuration of the software system mirrors the software system pictured in the baseline and the requirements documentation.

Software Configuration Auditing

- Asks the following questions:
  1. Has the specified change been made?
  2. Has a formal technical review been conducted to assess technical correctness?
  3. Has the software process been followed and standards been applied?
  4. Have the SCM procedures for noting the change, recording it, and reporting it been followed?
  5. Have all related SCIs been properly updated?
Software Configuration Auditing

Q: Why needed?
A: May help ensure a stopping point for a phase.

Software Configuration Status Accounting

- Allows to gather statistics about the system and the development process
  - Component with most changes
    => Question its quality?
- Provides a mechanism for maintaining a record of where the system is at any point with respect to what appears in published baseline.
  - It may take some time before a change is initiated or completed.
Software Configuration Status Accounting

- Why needed?
  - Ensure that there is progress within the development of the project.
  - Track updates to baselines.
  - Provide data for quality improvement

Requirements for Effective SCM

- **Repository:** shared DB for artifacts with controlled access to prevent overwrites.
- **Version management:** Maintain history of changes made to each artifact; provide ability to see how version was created.
- **Workspace control:** Private workspace with ability to check out from repository and check in with new version number.
SCM Tools

- Earlier: product oriented tasks, e.g., versions
  - SCCS, RCS
  - Concurrent Version System (CVS)
  - Subversion (SVN)
  - Git/GitHub
- Modern: more functionalities, e.g., process

Lock-Modify-Unlock System*

Copy-Modify-Merge System

Two users copy the same file

Harry

Sally

Repository

A

Read

A

Read

They both begin to edit their copies

Harry

Sally

Repository

A’

A’

A”

A”

Sally publishes her version first

Write

A”

Harry

Sally

Repository

Harry gets an “out-of-date” error

Write

A”

Harry

Sally

Copy-Modify-Merge System (Cont.)

Harry compares the latest version to his own

Repository

A”

Read

A”

Sally

A new merged version is created

Write

A”

A”

Harry

Sally

The merged version is published

A”

Now both users have each other’s changes

Repository

A”

Read

A”

Harry

Sally
Project Assignment

- Complete the configuration management plan for your team.
- Due date:
  - First Draft: Friday 09/8/2017
  - Final Draft: Friday 09/15/2017
- Leader: System Analyst

Quiz

- I wasn’t kidding on Tuesday
<table>
<thead>
<tr>
<th>Day/Time</th>
<th>Team</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday 10:00-11:00</td>
<td></td>
</tr>
<tr>
<td>Monday 11:00-12:00</td>
<td></td>
</tr>
<tr>
<td>Monday 2:00-3:00</td>
<td></td>
</tr>
<tr>
<td>Monday 3:00-4:00</td>
<td></td>
</tr>
<tr>
<td>Monday 4:00-5:00</td>
<td>T4</td>
</tr>
<tr>
<td>Wednesday: 10:00-11:00</td>
<td>T6</td>
</tr>
<tr>
<td>Wednesday: 11:00-12:00</td>
<td>T3</td>
</tr>
<tr>
<td>Wednesday: 2:00-3:00</td>
<td>T1</td>
</tr>
<tr>
<td>Wednesday: 3:00-4:00</td>
<td></td>
</tr>
<tr>
<td>Wednesday: 4:00-5:00</td>
<td>T2</td>
</tr>
<tr>
<td>Wednesday: 5:00-6:00</td>
<td>T5</td>
</tr>
<tr>
<td>Thursday: 10:00-11:00</td>
<td></td>
</tr>
<tr>
<td>Thursday: 11:00-12:00</td>
<td></td>
</tr>
</tbody>
</table>