Course Material No. 6

SOFTWARE ENGINEERING 2

FE LARWA-HABLANIDA
Course Instructor

RISK MANAGEMENT AND CONFIGURATION CONTROL 2

6

LEARNING OUTCOMES

At the end of the lesson, the learner will be able to:

- Explain the importance of risk management and configuration control in software projects.
- Identify, analyze, and prioritize risks using structured approaches.
- Describe the configuration management process and its role in maintaining system integrity.

RESOURCES NEEDED

For this lesson, you would need the following resources:

- PPT/Module
- Pencil and Paper

DISCUSSION:

INTRODUCTION TO CONFIGURATION CONTROL

Configuration Control is the process of systematically managing changes to a system, product, or project's configuration throughout its life cycle to ensure *consistency with requirements, traceability of changes,* and *integrity of the system or product.*

It is a key element of Configuration Management (CM).

- *Maintain accuracy of the product or system's design, functionality, and documentation.*
- *Ensure changes are authorized, documented, and communicated before implementation.*
- *Prevent uncontrolled changes that could lead to errors, delays, or cost overruns.*

Components of Configuration Control

Configuration Control is part of Configuration Management and ensures that all changes to a system's baseline are properly evaluated, approved, implemented, and documented.

1. **Change Identification**
   - Capturing and documenting proposed changes (Change Requests).
   - Includes details such as the reason for change, affected items, and expected impact.
   - A proposed change is documented (Change Request).
   - Includes reason, impact, and affected configuration items.
2. **Change Evaluation**
   - Assessing the technical, operational, cost, schedule, and risk impact of the proposed change.
   - Ensures that changes are feasible and aligned with project goals.
   - Assess the technical, cost, and schedule impact.
   - Risk analysis is performed.
3. **Change Approval/Disapproval**
   - A **Configuration Control Board (CCB)** or designated authority decides whether to approve, reject, or defer changes.

- Decisions are based on evaluation results and project priorities.

4. **Change Implementation**
   - Approved changes are planned, tested, and integrated into the system.
   - Ensures proper execution without disrupting system stability
5. **Change Verification & Validation**
   - Confirms that changes are correctly implemented and meet requirements.
   - Prevents introduction of errors or inconsistencies.
6. **Documentation & Status Accounting**
   - Updating configuration records, baselines, and change logs.
   - Maintaining traceability of all changes (who, what, why, when).
   - Provides visibility of current configuration state.
7. **Auditing & Review**
   - Conducting configuration audits (functional and physical) to ensure changes are properly documented and implemented.
   - Confirms consistency between design documents, baselines, and actual product/system.

CONFIGURATION MANAGEMENT PROCESS

Configuration Management (CM) is a structured approach to ensure that the functional, physical, and performance characteristics of a product, system, or service are consistent and well-documented throughout its life cycle. It ensures that changes are controlled and records are maintained so the current state is always known.

Configuration Identification

Configuration Identification is the process of **selecting, defining, and documenting the characteristics of Configuration Items (CIs)** that make up a system. It establishes **baselines** and ensures every component is uniquely identifiable and traceable throughout the lifecycle..

- **Activities**

1. **Selection of Configuration Items (CIs)**
   - Identify all components (hardware, software, documents, etc.) that need to be managed.
   - Criteria for selection: criticality, cost, risk, and impact.
2. **Defining Attributes of Each CI**
   - Name, version number, supplier, function, and dependencies.
   - Defines what makes one version different from another.

3. **Establishing Baselines**

A **baseline** is a formally approved and documented version of a product, system, or component at a specific point in time. It serves as a **reference point** against which future changes are controlled and measured.

a. **Functional Baseline** – initial requirements/specifications.

- Established after the requirements definition phase.
- Documents the system's functional and performance requirements.

Example:

- *System Requirement Specification (SRS).*

b. **Allocated Baseline** – system-level requirements distributed to subsystems.

- Established after system design is allocated to subsystems/components.
- Specifies requirements for each subsystem.

Example:

- *Subsystem specifications (hardware/software modules).*

c. **Product (or Design) Baseline** – final design ready for production or deployment.

- Established at the end of the **detailed design phase**.
- Documents the final design for production, coding, or construction.

Example:

- *Engineering drawings, source code, and database schema.*

4. **Unique Identification / Numbering System**
   - Assign labels, part numbers, document numbers, or software version numbers.
   - Ensures no ambiguity between different versions.
5. **Documentation**
   a. Maintain a **Configuration Item Record (CIR)** or **CI Register** with all details.
   b. Becomes the official reference point for future control and audits.

Configuration Control

It prevents **unauthorized or unplanned changes** that could cause inconsistencies, errors, or risks in the system. It manages changes to CIs in a **systematic and documented** way.

- **Activities:**
  - *Process changes requests.*
  - *Conduct impact assessments.*
  - *Approve or reject changes through a **Change Control Board (CCB)**.*

Example:

- *Approving a firmware update only after testing.*

Configuration Status Accounting

Configuration Status Accounting (CSA) is the process of **recording, tracking, and reporting information** about configuration items (CIs) and their changes throughout the system or product lifecycle.

- Maintain and report accurate records of the configuration's current state and history.
- **Activities:**
  - *Track versions, baselines, and changes.*
  - *Generate reports for stakeholders.*

Example:

- *A database showing which software version is installed on each workstation.*

Configuration Verification and Audit

Configuration Verification and Audit is the process of ensuring that a system's configuration items (CIs) and baselines are complete, consistent, and compliant with requirements, documentation, and approved changes.

- Confirm that the configuration matches its specifications and documentation.

➢ **Types:**

1. **Functional Configuration Audit (FCA)**
   - Verifies that the product meets functional and performance requirements from the Functional Baseline.
   - Verifies that functional requirements are met

   Example:

   - *Running test cases to confirm that a software module delivers all required features.*
2. **Physical Configuration Audit (PCA):**
   - Verifies that the as-built product matches the Product Baseline (drawings, code, design docs).
   - Verifies the product matches physical documentation.

   Example:

   - *Checking that delivered hardware matches engineering drawings, or that final code matches documented architecture.*

Documentation and Communication

Documentation and Communication in CM refers to the creation, maintenance, distribution, and sharing of configuration information to ensure that all stakeholders have accurate, current, and consistent knowledge of configuration items (CIs), changes, baselines, and status.

- Keep all relevant documentation up to date and ensure stakeholders are informed.
- **Activities:**
    - *Update design documents, manuals, and test reports.*
    - *Communicate changes to relevant teams.*

Example:

- *Updating the user manual after adding a new system feature.*

TOOLS AND TECHNIQUES FOR CONFIGURATION MANAGEMENT

Tools for Configuration Management (CM)

Tools for Configuration Management (CM). These tools support the identification, control, status accounting, and auditing of configuration items (CIs) across the system or software lifecycle.

1. **Version Control / Source Code Management Tools (SCM)**

Source Code Management (SCM) Tools are software systems that track, control, and manage changes to source code and related files. They enable multiple developers to collaborate, keep a history of changes, manage versions, and ensure traceability and rollback capability.

- Used to manage software versions, code changes, and history tracking.
- It uses Track code changes, manages branches, and ensures traceability of software development.

➢ **Types of SCM**

a. **Centralized Version Control Systems (CVCS)**

A **Centralized Version Control System (CVCS)** is a type of version control where there is one central repository that contains the official version of the project. Developers must connect to the central server to check out code, make changes, and commit updates.

- One central repository; developers check in/out code.

Examples:

- *CVS, Subversion (SVN), Perforce.*
- A company using **Subversion (SVN)** keeps the project's central repository on a server.
    - *Developers check out files, make changes, and commit them back to the central server.*
    - *If the server crashes or is inaccessible, no one can commit new changes until it is restored.*

b. **Distributed Version Control Systems (DVCS)**

A **Distributed Version Control System (DVCS)** is a type of version control where every developer has a full copy of the entire repository (including history) on their local machine.

- Each developer has a full copy of the repository, enabling offline work and easier collaboration.

Examples:

- *Git*
- *Mercurial*
- *Bazaar*

2. **Build and Release Management Tools**

Build and Release Management Tools are software solutions that help automate the process of compiling, packaging, testing, and deploying applications.

- Automate the process of building, integrating, and deploying systems.
- **Use:** Continuous Integration (CI) and Continuous Deployment (CD).

Examples:

- *Jenkins*
- *Maven*
- *Gradle*
- *Azure DevOps Pipelines*

3. **Configuration Control and Change Management Tools**

Configuration Control Tools and **Change Management Tools** are systems that help organizations:

- *Track, review, approve/reject, and implement changes to configuration items (CIs).*
- *Ensure that all changes are authorized, documented, and communicated.*
- *Provide visibility into the impact, status, and history of changes across the system or project.*

- Support **Change Requests (CRs), approval workflows, and baseline management**.
- **Use:** Track change approvals, enforce baselines, control system modifications.

Examples:

- *IBM Rational ClearCase*
- *Helix Core (by Perforce)*
- *Serena Dimensions CM*

4. **Issue Tracking and Collaboration Tools**

Issue Tracking and Collaboration Tools are software solutions that help teams log, manage, track, and resolve issues (such as bugs, feature requests, tasks, or change requests) while enabling effective communication and teamwork among stakeholders.

- Help manage **change requests, bug reports, and task assignments**.
- **Use:** Provide visibility into the status of changes and progress.

Examples:

- *Jira*
- *Trello*
- *Redmine*
- *Asana*

5. **Infrastructure as Code (IaC) / Automation Tools**

Infrastructure as Code (IaC) / Automation Tools are software solutions that enable organizations to define, provision, configure, and manage IT infrastructure using code, rather than relying on manual processes.

- Manage **system configurations and environments** automatically.
- **Use:** Ensure consistency of infrastructure, automate deployments, reduce manual errors.

Examples:

- *Ansible*
- *Puppet*
- *Chef*
- *Terraform*

6. **Configuration Status Accounting and Audit Tools**

Configuration Status Accounting (CSA) tools are software that record, track, and report the status of configuration items (CIs), changes, and baselines throughout the system's lifecycle.

Configuration Audit tools are software that **verify and validate** whether the actual configuration matches the documented and approved baseline (through Functional Configuration Audit and Physical Configuration Audit).

- Provide **traceability, reporting, and compliance checks**.
- **Use:** Maintain records of CIs, generate reports, and support audits.

Examples:

- *ServiceNow (for IT Configuration Management Database – CMDB)*
- *BMC Helix*

- *Micro Focus CM tools*

Techniques for Configuration Management (CM)

Techniques for Configuration Management (CM) are the methods and practices used to effectively implement CM across projects, systems, or software.

1. **Baseline Management**
   - Establish and maintain **reference points (baselines)** for requirements, design, and product.
   - Prevents uncontrolled changes.

   Example:
   - *Functional, Allocated, and Product Baselines in System Development.*
2. **Change Control**

   It ensures that every change is identified, evaluated, approved/rejected, implemented, verified, and documented in a controlled manner.

   - Formal process for handling Change Requests (CRs).
   - Involves identification, evaluation, approval, implementation, and documentation.
   - Usually managed by a Configuration Control Board (CCB).

   Example:
   - *A client requests an extra feature in a software project.*
   - *Developer raises a Change Request.*
   - *Project team evaluates cost, effort, and risks.*
   - *CCB approves → team implements → QA verifies → documentation updated → baseline revised.*
3. **Version Control**

   Version Control is the process of tracking, managing, and controlling different versions of configuration items (CIs) such as software code, documents, hardware designs, or system components. Maintaining and tracking different versions of software, documents, or hardware.

   - Ensures traceability and rollback if needed.
   - Tools: Git, SVN, Mercurial.

   Example:
   - *A team is developing a mobile app.*
   - *Developer A works on the login feature, Developer B on payment.*
   - *Both create separate branches in Git.*
   - *Later, their changes are merged into the main branch.*
   - *The release is tagged as v2.0 for deployment.*

4. **Configuration Audits**

Configuration Audits are formal reviews conducted to verify that configuration items (CIs) and baselines are complete, correct, and compliant with their requirements and documentation.

- Confirms accuracy and completeness of changes.

5. **Configuration Status Accounting (CSA)**

Configuration Status Accounting (CSA) is the process of recording, tracking, and reporting information about configuration items (CIs) and changes throughout the product or system lifecycle.

Example:

- *In a software project:*
  - *CSA shows that Module A is at Version 2.1, linked to Change Request #045 (implemented).*
  - *Module B is still at Version 1.3, pending approval for an update.*
  - *The report highlights the Product Baseline v2.0 as the current release.*

6. **Documentation Management**

Documentation Management is the process of creating, organizing, controlling, updating, and maintaining project/system documents so they always reflect the current approved configuration.

It ensures that documentation is accurate, consistent, traceable, and accessible throughout the product or system lifecycle.

- Maintaining accurate, updated, and accessible documentation of configuration items (CIs).
- Supports audits, traceability, and training.

7. **Automation and Tool Support**

Automation and Tool Support in Configuration Management refers to the use of specialized software systems and scripts to automate CM activities such as version control, builds, testing, deployment, change management, and reporting.

- It reduces manual effort, minimizes errors, and ensures faster, more reliable processes.
- Using tools like Jira, Git, Jenkins, Ansible to automate
  - *Version tracking*
  - *Build and deployment*
  - *Change approval workflows*
- Reduces human errors and speeds up processes.

Example:

- *A software update is committed to GitHub.*
- *Jenkins automatically builds and tests the new version.*
- *If tests pass, Ansible deploys the update to servers.*
- *Jira automatically updates the change request status to "Implemented."*
- *ServiceNow generates a status report for management.*

8. **Configuration Item (CI) Identification and Tagging**

Configuration Item (CI) Identification and Tagging is the process of selecting, defining, and uniquely labeling items (hardware, software, documents, or processes) that need to be controlled under the Configuration Management (CM) process.

- Assigning unique identifiers, labels, or tags to configuration items.
- Helps with tracking across lifecycle stages.
- Each is stored in the CM system with unique tags.

Example:

- *Software release tags (v1.0, v1.1), hardware serial numbers.*
- *A software project identifies the following CIs:*
  - *SRS-2025-v1.0 (Requirements Document)*
  - *UI_Module_v2.3 (Software Component)*
  - *DBSchema_v1.2 (Database Schema)*
- *When a change request affects the UI, the team can trace exactly which version of UI_Module is impacted.*

9. **Risk-Based CM**

Risk-Based Configuration Management is the practice of prioritizing CM activities (identification, control, audits, documentation, etc.) based on the level of risk or criticality of configuration items (CIs).

- Prioritizing configuration control on high-risk or high-impact items.
- Helps optimize resources by focusing on critical components.

Example:

- *In an* **Aerospace Project***:*
  - *The flight control software is high risk → strict CM (formal reviews, baseline audits, strict versioning).*
  - *The user manual is low risk → lightweight document control.*
- *In an* **IT Project***:*
  - *The database schema (high risk) → requires strong version control and approval.*
  - *The UI color theme (low risk) → minimal CM effort.*

REFERENCES

- Pressman, Roger S, Software Engineering: A Practitioner's Approach, 9th Edition, Published by Mc Graw-Hill (2019)
- Sommerville, Ian, Software Engineering 10th Edition, Published by Pearson Education, Inc (2016)
- Bass, Len, Clements, Paul, & Kazman, Rick., Software Architecture in Practice (3rd Edition). Addison-Wesley, 2012.