Course: Software Configuration Management

Week 15: Course Review

Lecturer: Yimer Amedie (MSc.)
Addis Ababa Science and Technology University, Ethiopia

November, 2025

Contents

Figure 1: Concepts of SCM
(Source: OpenAI, 2025)

- Introduction of the fundamental concepts of SCM
- Core SCM process areas
- Integration of SCM practices within SDLC
- SCM Plans, change management
- Current trends, challenges and best practices
- Summary

Learning Outcomes

After completing this lesson, you will be able to:

- Summarize the key concepts of software configuration management.
- Analyze how core areas such as identification, control, status accounting and auditing applied in software projects.
- Reflect on your learning progress and identify key takeaways for future application in real-world projects.
- Prepare effectively for final assessments or project presentations

Introduction

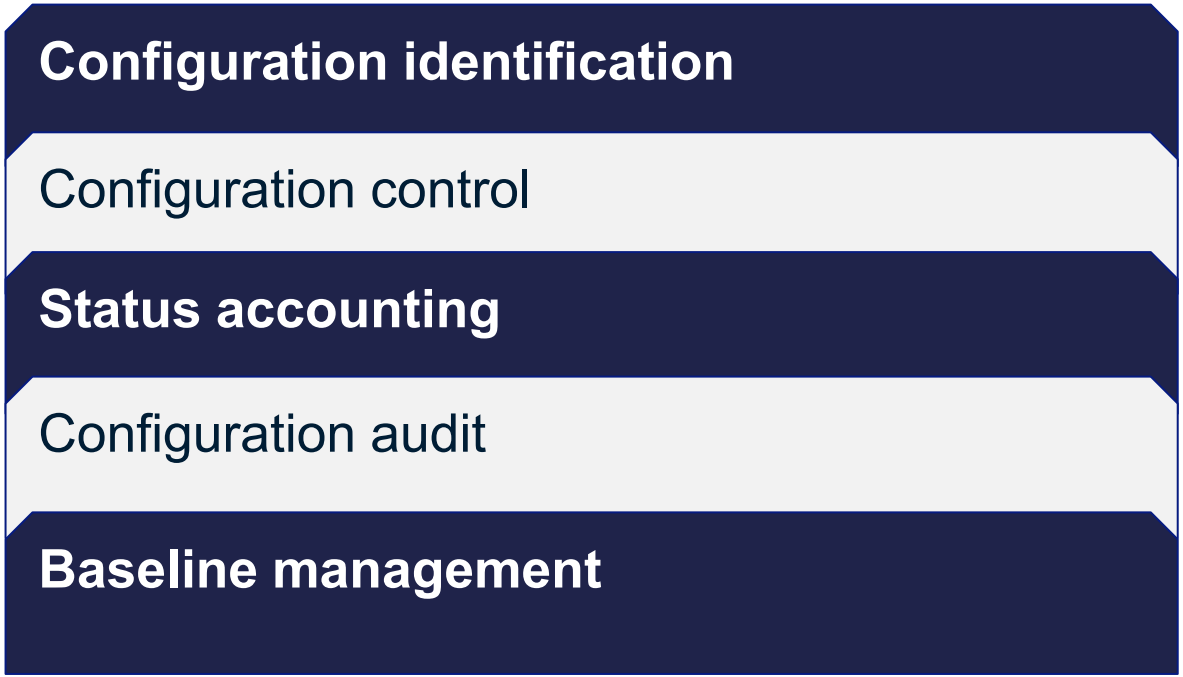
- SCM is the discipline of systematically controlling software changes and maintaining the integrity of evolving systems (Leon, 2015).
  - Ensures consistency and reproducibility
  - Tracks configuration items
  - Supports collaboration
  - Basis for quality management

Why SCM is important?

- SCM plays a critical role in ensuring software projects remain organized and predictable.
  - Reduces risk and confusion
  - Supports collaboration
  - Enables controlled releases
  - Improves traceability
  - Protects deliverables

Core Components of SCM

- Every SCM system is built around five core components that define how changes are managed.



Core Components of SCM

Configuration identification

Determines which items need control, giving them unique names and version labels

- Define configuration items (CIs)
- Assign unique identifiers
- Establish naming conventions
- Create baselines
- Document relationships

Core Components of SCM

Configuration control

Manages how changes to those items are proposed, evaluated, and approved

- Manage changes systematically
- Use change control boards (CCB)
- Apply formal approval workflows
- Track implementation status
- Enforce rollback procedures

Core Components of SCM

Status accounting

Records the history and current state of each configuration item, ensuring traceability across time

- Record configuration item history
- Maintain version and change logs
- Generate reports and dashboards
- Provide traceability for audits
- Support decision-making

Core Components of SCM

Configuration audit

Checks that all items match their documentation and functional requirements

- Verifies configuration items meet requirements and standards.
- Ensures documentation matches the delivered product.
- Involves functional and physical audits.
- Identifies discrepancies and corrective actions.
- Enhances product integrity and readiness for release.

Core Components of SCM

Baseline management

Ensure that approved versions of the system are preserved as stable reference points for future development

- Baselines represent approved versions of configuration items.
- They act as reference points for future development.
- Version control manages changes systematically.
- Tools like Git, SVN, and Mercurial are commonly used.
- Branching and merging support parallel development efforts

Core Components of SCM

- **Configuration Items (CIs):**
  - The building blocks of configuration identification
  - A uniquely identifiable component of a system managed through SCM
  - Types


Software


Documents


Test Data


Libraries and tools

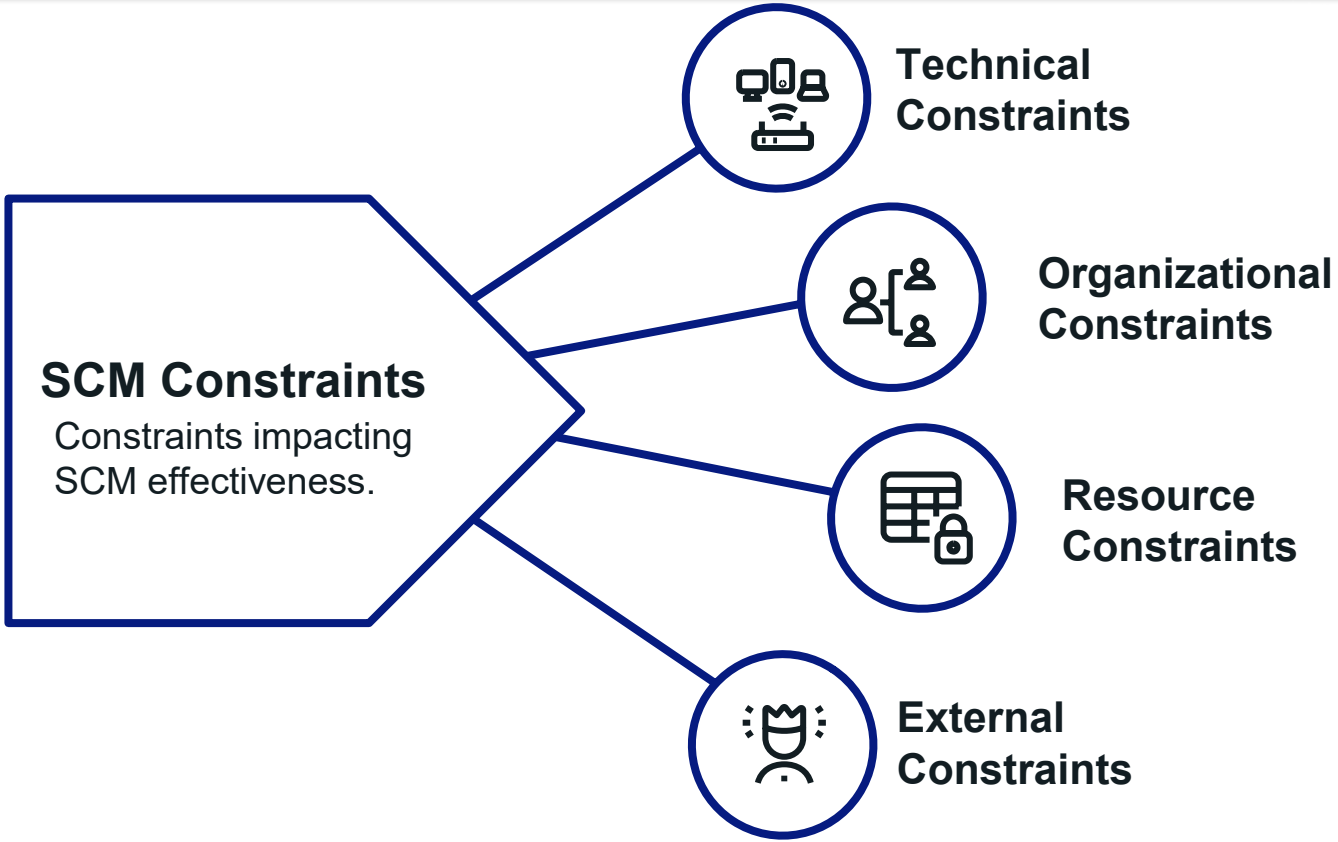
SCM in Project lifecycle

- SCM plays a vital role throughout the entire project lifecycle.
  - Applies from initiation to closure
  - Integrates with requirements and design
  - Supports build, test, and release phases
  - Maintains traceability across stages
  - Facilitates maintenance and support

Organizational Context

- **Context:** Environment where SCM is practiced.
- Includes
  - Culture, structure, and leadership
- Defines how decisions are made
- Affects resource allocation and priorities
- Shapes SCM effectiveness and adoption
- How SCM fit in to
  - Centralized Vs Decentralized structure

SCM Constraints



SCM Planning and Documentation

- SCM planning and documentation form the foundation of effective configuration management.
  - SCM planning defines scope, roles, and policies.
  - The SCM Plan (SCMP) guides all SCM-related activities.
  - It ensures consistent handling of configuration items.
  - Documentation supports traceability and accountability.
  - Regular updates keep the plan relevant and effective.

Components of SCM Plan

According to (IEEE, 2012), the SCM should have the following sections:

- **Introduction:**
  - Purpose, scope, organizational relationships, references
- **Criteria for Identification:**
  - Configuration Items (CIs) covered by CM
- **Limitations & Assumptions:**
  - Factors affecting the plan
- **CM Responsibilities & Authorities:**
  - Roles, responsibilities, authority lines

Components of SCM Plan … (2)

According to (IEEE, 2012), the SCM should have the following sections:

- **Project Organization:**
  - Structure and CM interfaces
- **Applicable Policies & Procedures:**
  - Standards, directives, methods
- **Planned Activities:**
  - Schedules, resources, milestones
- **CMP Maintenance:**
  - Update and review mechanisms

Change Management in SCM

- Change management is the backbone of controlled evolution in software projects.
  - Change management ensures systematic modification control.
  - Each change passes through defined review and approval stages.
  - The Change Control Board (CCB) manages major changes.
  - Documentation of changes ensures transparency and traceability.
  - Effective change management reduces project risks.

SCM in Agile Development

- In Agile development environments, SCM must adapt to fast-paced, iterative workflows.
  - Agile SCM supports frequent, iterative software updates.
  - Configuration control aligns with short sprint cycles.
  - Continuous integration maintains consistency and visibility.
  - Lightweight documentation replaces heavy formalities.
  - Collaboration between developers and testers is key.

SCM in DevOps Environments

- In a DevOps environment.
  - SCM integrates development and operations processes.
  - Supports Continuous Integration (CI) and Continuous Delivery (CD).
  - Automation ensures faster, reliable deployments.
  - Tracks infrastructure and code together.
  - Enables rapid feedback and rollback mechanisms

SCM Automation Tools

- Automation transforms SCM from a manual, error-prone activity into an efficient and reliable process.
  - Automation enhances efficiency and consistency in SCM.
  - Tools include Git, Jenkins, Docker, Ansible, and Terraform.
  - Automates builds, tests, and deployments.
  - Reduces human errors and manual interventions.
  - Integrates with CI/CD pipelines for faster delivery

SCM and Compliance Management

- Compliance and security are integral aspects of SCM.
  - SCM ensures adherence to regulatory and security standards.
  - Tracks all configuration changes for audit readiness.
  - Supports data protection and risk mitigation.
  - Automation enforces consistent security configurations.
  - Ensures accountability through traceable version history.

Types of Compliance in SCM

Regulatory Compliance

Adherence to legal and industry rules

Contractual Compliance

Fulfillment of customer agreements

Standards Compliance

Alignment with ISO, IEEE,CMMI standards

Internal Compliance

Following company policies and processes

Ethical Compliance

Upholding responsibility and transparency

Release Management

- Serves as the bridge between development and operations.
  - Coordinates the deployment of approved configurations to production.
  - Ensures that all changes are tested, validated, and approved before release.
  - Maintains version integrity across environments
    - development, testing, production).
  - Uses automated pipelines and CI/CD tools for faster, reliable delivery.
  - Provides documentation and rollback strategies for controlled releases

Emerging Technologies in SCM

- Emerging technologies are reshaping the way SCM operates.
  - AI assists in predictive analysis and anomaly detection.
  - Blockchain supports immutable configuration history.
  - Cloud-native SCM tools enhance scalability and collaboration.
  - Containerization (Docker, Kubernetes) simplifies environment control.
  - Integration with CI/CD pipelines increases automation

Challenges in SCM Implementation

- While SCM offers significant benefits, implementing it effectively can be challenging.
  - Resistance to process changes within teams.
  - Complexity in managing multiple versions and branches.
  - Inadequate tool integration and automation.
  - Security and compliance gaps in configurations.
  - Lack of skilled professionals and clear policies

Configuration Drift Management

- Configuration drift refers to unintended differences between system environments.
  - It occurs when environments become inconsistent.
  - Common in cloud and distributed systems.
  - Automated monitoring detects and resolves drifts.
  - Tools like Ansible, Chef, and Puppet help manage drift.
  - Preventing drift improves stability and security

SCM Best Practices

- Implementing best practices is critical for achieving efficient and reliable SCM.
  - Define clear configuration item identification policies.
  - Maintain regular audits and reviews for accuracy.
  - Automate builds, testing, and deployment processes.
  - Foster collaboration between development and operations.
  - Keep documentation updated and accessible.

Summary

- SCM ensures control, stability, and reliability in software systems.
- Acts as a bridge between development, operations, and quality assurance.
- Supports modern delivery models like Agile and DevOps.
- Encourages accountability and continuous improvement.
- Remains essential for long-term software success.

Summary … (2)

- Introduced the purpose, scope, and benefits of Software SCM
- Covered core SCM activities:
  - Identification, control, status accounting, and auditing.
- Discussed SCM organizational planning, policies & process integration.
- Explored SCM planning, tool selection & automation best practices.
- Covered software release management, delivery, and deployment strategies.

References

1. Leon, A. (2015). Software Configuration Management Handbook (3rd ed.). Norwood: Artechhouse. Retrieved September 4, 2025.
2. OpenAI. (2025, September 4). SCM history and concepts [AI-generated image]. ChatGPT (Sora). https://chat.openai.com/
3. IEEE. (2012). Standard for configuration management in systems and software engineering (IEEE Std 828-2012). Retrieved September 24, 2025 from https://raw.githubusercontent.com/Orthant/IEEE/master/828-2012.pdf

Thank You!

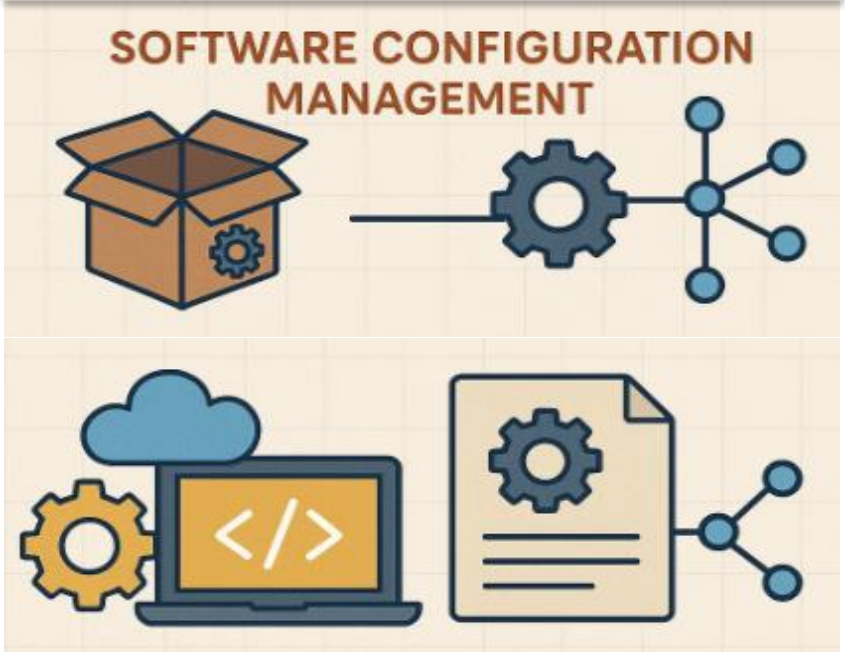


Figure 2: Concepts of SCM (Source: OpenAI, 2025)