**Experiment No. 11: Case Study on Software Configuration Management (SCM)**

### **Aim:**

To study and analyze the principles, processes, and tools of software configuration management (SCM) by exploring a case study.

### **Theory:**

Software Configuration Management (SCM) is a critical process in software engineering that ensures consistency, traceability, and control over software components and artifacts during development. It includes managing changes, maintaining integrity, and preventing issues such as version conflicts.

**Key Concepts of SCM:**

1. **Configuration Items (CIs):** Files, code modules, documents, or artifacts under version control.
2. **Version Control:** Managing multiple versions of files or code. Examples: Git, SVN.
3. **Change Management:** Recording and controlling modifications in the project.
4. **Build Management:** Automating the creation of executable software from source code.
5. **Release Management:** Delivering the final software to stakeholders.

**Importance of SCM:**

* Maintains consistency and integrity across project components.
* Reduces development time by providing structured processes.
* Facilitates team collaboration by tracking and merging changes.
* Enhances software quality through better traceability and control.

**Popular SCM Tools:**

* **Git:** Distributed version control for tracking source code changes.
* **SVN:** Centralized version control system.
* **Jenkins:** Automation tool for continuous integration and delivery.
* **Docker:** Manages containerized environments.

### **Learning Objectives:**

1. To understand the importance of SCM in software projects.
2. To explore tools and techniques used in SCM.
3. To analyze a real-world case study on SCM and its implementation.

### **Learning Outcomes:**

At the end of this experiment, students will be able to:

1. Describe the processes and components of software configuration management.
2. Analyze the application of SCM in a real-world scenario.
3. Evaluate the effectiveness of SCM tools and techniques.

### **Course Outcomes (COs):**

* **CO5:** Demonstrate an understanding of modern tools and processes in software engineering, including SCM.

### **Cognitive Levels of Attainment as per Bloom’s Taxonomy:**

* **L2 (Understand):** Explain the processes and benefits of SCM.
* **L4 (Analyze):** Evaluate SCM implementation in a real-world scenario.
* **L5 (Evaluate):** Assess the effectiveness of SCM tools and practices.

### **Programme Outcome (POs):**

* **PO1:** Apply knowledge of engineering fundamentals to software management practices.
* **PO5: Use modern tools and techniques for software development and management.**
* **PO6: Analyze societal and environmental impacts of software practices.**
* **PO7: Understand the sustainability of software systems through effective management.**
* **PO11: Apply project management principles to SCM.**

### **Programme Specific Outcome (PSOs):**

* **PSO1:** Utilize configuration management tools to ensure consistency in software solutions.

### **Result & Discussion:**

* **Result:** A case study on SCM was analyzed to identify best practices, challenges, and outcomes in the implementation process.
* **Discussion:**
  + Insights into the importance of version control and change management were gained.
  + The selected SCM tool's impact on software development and team collaboration was discussed.

### **Conclusion:**

This experiment provided an understanding of SCM processes and tools. By analyzing a case study, students learned how SCM ensures project integrity, reduces risks, and improves software quality.

### **Steps for the Experiment**

#### ****1. Select a Case Study on SCM:****

* Choose a real-world example where SCM was implemented effectively.
* Example case studies:
  + SCM practices in an open-source project using Git.
  + SCM in a corporate environment using Jenkins for continuous integration.

#### ****2. Analyze the Case Study:****

* Identify the key components of SCM implemented in the project:
  + Version control, change management, build automation, release management.
* Understand the challenges faced and solutions applied in the project.
* Evaluate the impact of SCM tools on the development process.

#### ****3. Document the Findings:****

* Write a detailed report covering the following sections:
  + **Introduction to the project**
  + **SCM processes implemented**
  + **Tools and technologies used**
  + **Challenges and resolutions**
  + **Outcomes and lessons learned**

### **Example Questions for Case Study Analysis**

1. **Describe the role of version control in the selected case study.**
2. **What SCM tools were used, and why were they chosen?**
3. **What were the key challenges faced during SCM implementation?**
4. **Evaluate the effectiveness of the SCM process in ensuring project success.**

### **Sample SCM Process in Git**

**Steps for SCM using Git:**

1. Initialize a repository: git init
2. Add files to version control: git add
3. Commit changes: git commit -m "Initial commit"
4. Create branches for feature development: git branch feature\_x
5. Merge branches after testing: git merge feature\_x
6. Use tags for releases: git tag v1.0

**Steps for the Experiment – Fair Share**

**1. Select a Case Study on SCM:**  
Selected Project: **Fair Share – A Full-Stack Expense Splitter Web App**  
Context: SCM practices implemented throughout the development using **Git**, **GitHub**, **Jenkins**, and **Docker** for CI/CD and environment management.

**2. Analyze the Case Study:**

**SCM Components Implemented:**

* ***Version Control****:* All source code maintained under Git with a structured branching strategy (main, dev, feature branches).
* ***Change Management****:* Pull requests with code reviews used to ensure changes are traceable and tested before merging.
* ***Build Automation****:* Jenkins used to automate testing and deployment of backend services.
* ***Release Management****:* Tags used in GitHub for stable releases and Docker used to manage deployment environments.

**Challenges Faced & Solutions:**

* *Merge Conflicts:* Solved by enforcing smaller, frequent commits and structured branching.
* *Environment Inconsistencies:* Resolved by containerizing services using Docker.
* *Testing Before Deployment:* Automated unit tests and builds integrated into Jenkins pipeline.

**Impact of SCM Tools:**

* Improved team coordination via Git and GitHub.
* Minimized deployment errors through CI/CD.
* Better traceability of changes and enhanced debugging with version history.

**3. Document the Findings:**

**Introduction to the Project:**  
Fair Share is a full-stack web application for managing and splitting group expenses with features like multi-currency support, debt management, and recurring bills.

**SCM Processes Implemented:**  
Git-based version control, change tracking via GitHub pull requests, Jenkins for CI/CD, Docker for consistent environments, and tagging for releases.

**Tools and Technologies Used:**  
Git, GitHub, Jenkins, Docker, Prisma, Node.js, PostgreSQL, React.

**Challenges and Resolutions:**  
Merge conflicts (solved via Git discipline), inconsistent environments (solved using Docker), testing before deployment (automated using Jenkins).

**Outcomes and Lessons Learned:**  
SCM ensured consistent project builds, smooth collaboration, and high-quality deliverables. Tools like Git and Jenkins were crucial in maintaining version history and automating deployment.

**Questions for Fair Share Case Study Analysis**

1. **Describe the role of version control in the selected case study:**  
   Git helped track all code changes, enabled collaboration with feature branches, and ensured rollback capability in case of bugs.
2. **What SCM tools were used, and why were they chosen?**  
   Git and GitHub for distributed version control, Jenkins for automated builds, and Docker for reproducible development environments.
3. **What were the key challenges faced during SCM implementation?**  
   Merge conflicts and environment mismatches. Solved using structured Git strategy and Docker.
4. **Evaluate the effectiveness of the SCM process in ensuring project success:**  
   SCM tools ensured stable releases, reduced human errors in deployment, and improved collaboration. SCM significantly contributed to the maintainability and scalability of Fair Share.

**SCM Process in Git used:**

1. Initialize a repository: git init
2. Add files to version control: git add .
3. Commit changes: git commit -m "Initial setup"
4. Create feature branches: git checkout -b feature/group-events
5. Merge branches after review: git merge feature/group-events
6. Tag release: git tag v1.0.0