Course: Software Configuration Management

Week 16: Final Exam – Answer Keys

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

November, 2025

Answer Keys

Part I: Multiple Choice ➔ 60%

Part II: Short Answer ➔ 15%

Part III: Case Analysis ➔ 25%

Total ➔ 100%

Multiple Choice – Answer Keys

Q. No.	Answer	Q. No.	Answer	Q. No.	Answer
1	B	11	B	21	A
2	B	12	B	22	B
3	B	13	C	23	B
4	A	14	A	24	C
5	B	15	B	25	B
6	C	16	B	26	D
7	B	17	B	27	D
8	C	18	C	28	A
9	C	19	B	29	B
10	B	20	A	30	C

Shor Answer – Answer Keys

1. Software Configuration Management (SCM) is a systematic process for identifying, organizing, and controlling changes in software to maintain consistency and integrity throughout the development lifecycle.
   Its main goal is to ensure that all software artifacts — such as code, documents, and configurations — are version-controlled, traceable, and reproducible, preventing confusion and errors caused by uncontrolled changes.
2. The four core SCM processes are:

- **Configuration Identification:** Defines and labels configuration items (CIs) and baselines for control.
- **Change Control:** Manages changes through formal evaluation and approval processes.
- **Configuration Status Accounting:** Records and reports the status and history of configuration items and changes.
- **Configuration Auditing:** Verifies that items conform to their specifications and approved baselines.

Shor Answer – Answer Keys … (2)

3. A Configuration Item (CI) is any component or artifact that must be managed under configuration control. It can be hardware, software, document, or process-related.

   Examples:

   - ✓ A source code file (e.g., login.js)
   - ✓ A requirements specification document

4. Automation reduces manual errors, speeds up repetitive tasks, and ensures consistent application of SCM policies. Tools like Git, Jenkins, and Ansible automate versioning, builds, testing, and deployment, improving traceability, audit readiness, and efficiency while supporting continuous integration and delivery.
5. Release management ensures that only approved and tested configurations are delivered to production environments in a controlled manner. It coordinates deployment, minimizes risks, supports rollback when needed, and guarantees that all releases are traceable, documented, and aligned with project objectives.

Case Analysis – Answer Keys

1. The conflict should be resolved through version control mechanisms by identifying the differences, discussing with both contributors, and merging changes into a consistent version. The updated version should be reviewed and committed as a new baseline following SCM change control procedures.
2. SCM should enforce controlled release procedures and approval workflows where only authorized personnel can deploy approved baselines. The incident should trigger an audit and corrective action, updating the SCM plan to strengthen controls.
3. Configuration drift can be prevented through automated configuration management tools (e.g., Ansible, Puppet) and regular synchronization checks. SCM should maintain version-controlled infrastructure scripts and perform audits to ensure environment consistency.

Case Analysis – Answer Keys … (2)

4. This issue indicates a failure in **configuration identification and change control**. Every change, regardless of size, must be linked to a formally approved requirement or change request. SCM should enforce strict traceability rules so each code change references an approved item in the requirements baseline. The team must update the SCM Plan to ensure future changes are tracked through change requests and that traceability matrices are maintained between requirements, design, code, and tests.
5. The issue shows poor **release management and configuration status accounting**. SCM must ensure that all released versions are properly labeled, stored, and retrievable from the repository. Each release should have associated documentation, including version identifiers, release notes, and rollback plans. To prevent recurrence, SCM should automate release packaging, maintain backup baselines, and test rollback procedures during release rehearsals. Proper auditing should confirm that rollback versions exist and are verified before any deployment.

The End!