BACHELOR OF COMPUTER APPLICATIONS

SEMESTER 5

DCA3103

SOFTWARE ENGINEERING

Unit 7

Configuration Management

Table of Contents

1. INTRODUCTION

The process of defining, recording, and reporting the status of the Configuration Items (CI) within a system is known as configuration management. It focuses on managing the various components and configurations of a software system throughout its lifecycle. It involves identifying, organizing, controlling, and tracking these components to ensure consistency, integrity, and traceability.

The main goal of Configuration Management is to establish and maintain a controlled and well-documented software development environment. It helps in keeping track of software versions, managing dependencies, and ensuring that all the required components and configurations are in place for successful software development and deployment.

Change management, version and release management, software maintenance, software reengineering, and software refactoring are just a few of the vital areas of software engineering where configuration management is essential.

1.1 Learning Objectives

After studying this unit, learners will be able to:

- *Identify the key components and configurations of a software system.*
- *Understand the importance of version control in managing software components.*
- *Analyse a software system and propose reengineering strategies to improve its performance or maintainability.*
- *Apply Configuration Management principles to manage changes in a software system.*

2. CHANGE MANAGEMENT

Change management evaluates and plans the change process. It makes sure that a change is made in the most efficient way possible. It ensures that the proper procedures are followed. This in turn ensures the quality and continuity of IT services of the organization.

2.1 Elements Of Change Management

Change management is the correct application of tools, skills, processes, and principles required for making a suitable change.

The three main elements of change management are:

i. Set of tools, skills, processes, and principles.
ii. Acceptance of change by the people.
iii. Achievement of the desired results of the project.

Let us now see these elements.

i. Set of tools, skills, processes, and principles

Change management uses several tools to bring about a successful change. A communication plan and a training plan are two of the most widely used tools. The change management process describes the sequence of activities that should be followed for a particular project. Generally, there are three phases. They are:

- Preparing the change
- Managing the change
- Reinforcing the change

Change management uses the skill of almost everyone in the organization. However, the contribution of executives, managers, leaders, and supervisors is more important. Change management uses certain principles for proper implementation.

ii. Acceptance of change by the people

When a change is made in an organization, the job of every individual changes. Changes that make an impact on the jobs of people need structure and planning to address the change of the people involved. An important aspect of managing changes that impact employees is to understand how everyone successfully makes the change. The success of a change depends on the adaptability of the individuals involved.

iii. Achievement of the desired results of the project

Having a change management process increases the probability of a project achieving its objectives on a large scale. It also increases the probability of maintaining the schedule and budget. This clause is important while discussing change management with executives, managers, leaders, and supervisors.

2.2 Objectives Of Change Management

The objectives of change management include the following:

- Ensure that the reason for the change is clear.
- Identify the important people who will be involved in the change activities.
- Define the sponsorship, communication, and involvement to all the stakeholders.
- Plan the activities and involvement of change sponsors.
- Plan how and when a certain activity should be carried out.
- Deliver appropriate messages.
- Assess the impact on the people and the organization.
- Plan activities that take care of the impact of the change.
- Ensure that people who are involved and are affected by the change understand the change properly and have proper guidance during this time.
- Assess the training needs due to the change and plan how this will be implemented.
- Have success indicators and ensure these are measured regularly.

2.3 Issues Of Change Management

Some of the common issues faced by change management are:

- ***Technology*** – A technology change like software change, new hardware, change in the operating system or even a procedural adjustment causes discomfort and problems for many. The IT departments generally implement changes with limited considerations for training and how people may react to the change.
- ***Mergers or acquisitions*** **–** If the company purchases or starts a new business, it may create doubts in the mind of the existing employees about their career. They will fear all kinds of changes that may happen. Generally, mergers or acquisitions bring in changes such as a change of role, leadership, or even termination.
- ***New leadership*** **–** The change of CEO or leadership departments compels the staff to make adjustments that may not be easy. If the person has been holding the position for a long time the adjustment becomes even more difficult.
- ***Staff changes*** **–** The addition or removal of members changes the dynamics of the team and the organization.
- ***Office relocation*** **–** It is not important if one is moving nearby or far away. The important aspect is ensuring that employees know the new location and assurance that the required facilities are available. This helps them to work with a relaxed mind.

3. VERSION AND RELEASE MANAGEMENT

Version and Release management are important aspects of Configuration Management in software engineering. They involve the management and control of different versions of software components and the packaging and deployment of software releases to end-users.

3.1 Version Management

Version management focuses on the control and tracking of different versions or iterations of software components within a software system. It involves assigning unique identifiers or labels to each version of a component and maintaining a history of changes made to these versions. Version management helps in keeping track of different iterations of the software components and their associated changes, enabling developers to understand and manage the evolution of the software system.

The principal tasks of version management include:

- ***Version Identification:*** Identifying and labelling different versions of software components to distinguish them from one another.
- ***Version Control:*** Tracking changes made to each version and managing the relationship between different versions.
- ***Version Baselines:*** Defining specific versions as baselines, which serve as reference points for future development or for creating software releases.
- ***Version Documentation:*** Documenting the changes made to each version and maintaining a version history for reference.

3.2 Release Management

Release management is concerned with the packaging, distribution, and deployment of software releases to end users. It involves coordinating the release process, ensuring that all necessary components and configurations are included, and managing the release documentation. Release management aims to deliver a stable and tested version of the software to users, considering factors such as compatibility, dependencies, and release notes.

The main tasks of release management are:

- ***Release Planning:*** Defining the scope and objectives of the release, determining the components to be included, and establishing release criteria and timelines.
- ***Release Packaging:*** Assembling all the required software components, documentation, and configurations into a deployable package or release build.
- ***Release Testing:*** Conduct thorough testing of the release to ensure its quality and compatibility with the target environment.
- ***Release Deployment:*** Distributing and installing the release package on target systems, following predefined deployment procedures.
- ***Release Documentation:*** Creating and maintaining release notes, user manuals, and other documentation to provide instructions and information to end-users.

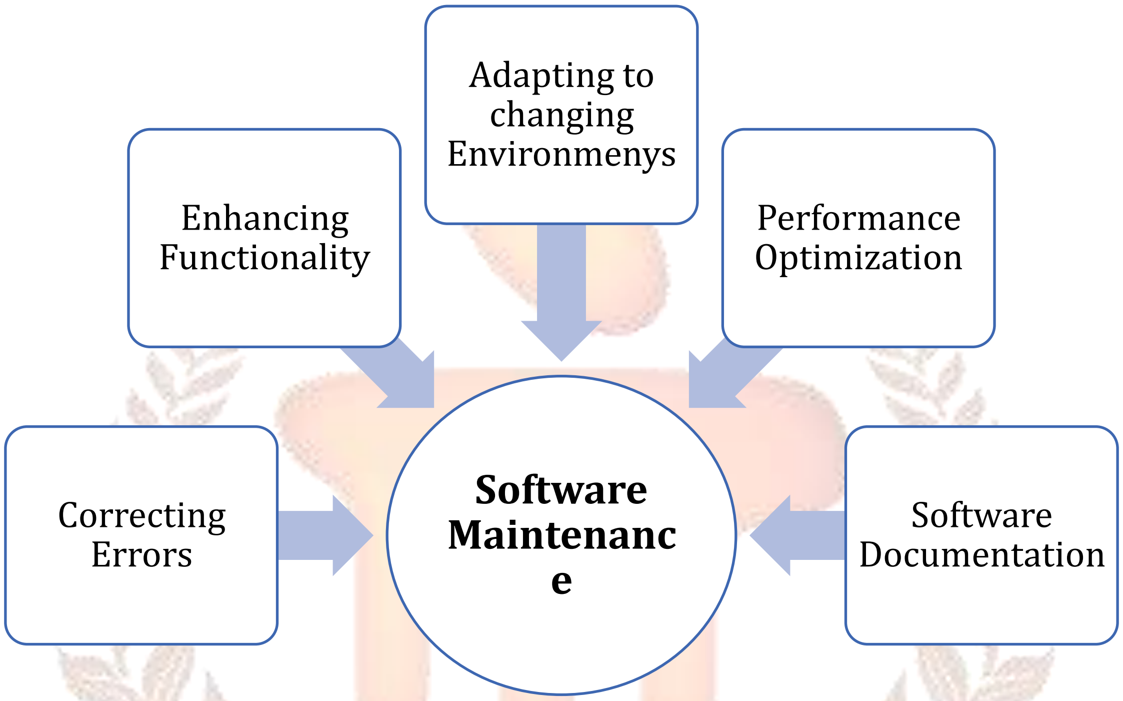
4. SOFTWARE MAINTENANCE

Software In most of the field's maintenance, software maintenance is bout repairing things that break or wear out. But in the software field, the nature of software is such that it never breaks or wears out. Software is an intangible thing, having no physical form, so there is nothing to break or wear out. The software can have errors in it. The software can be modified to do new things. Software errors are not due to material weakness, but rather to errors made when the software was being built or errors made as the software is being changed. So, software maintenance is about fixing those errors as they are discovered and making those changes as they become necessary.

Software maintenance encompasses various tasks, including:

1. **Correcting Errors:** Software maintenance involves identifying and fixing errors or defects in the software. These errors may manifest as bugs, malfunctions, or unexpected behaviour. The goal is to eliminate these errors to ensure the software functions as intended.
2. **Enhancing Functionality:** Software maintenance also includes adding new features or functionalities to the software to meet changing user requirements or business needs. This can involve modifying existing code or adding new code modules to extend the software's capabilities.
3. **Adapting to Changing Environments:** Software systems often need to adapt to changes in the operating environment, such as new hardware or software platforms, operating system upgrades, or changes in regulatory or security requirements. Maintenance activities are carried out to ensure the software remains compatible and performs optimally in the evolving environment.
4. **Performance Optimization:** Maintenance may involve analysing and optimizing the performance of the software system. This can include identifying bottlenecks, optimizing algorithms, or improving resource utilization to enhance the software's efficiency and responsiveness.
5. **Software Documentation:** Documentation maintenance is an important aspect of software maintenance. It involves updating and enhancing the documentation to reflect

changes made to the software, making it easier for developers and users to understand and work with the system.



5. SOFTWARE RE-ENGINEERING

Software reengineering refers to the process of modifying or transforming an existing software system while managing its configuration and ensuring the integrity of the software components.

The reengineering paradigm shown in Figure 7.1 is a cyclical model. This means that each of the activities presented as a part of the paradigm may be revised. For any cycle, the process can terminate after any one of these activities.

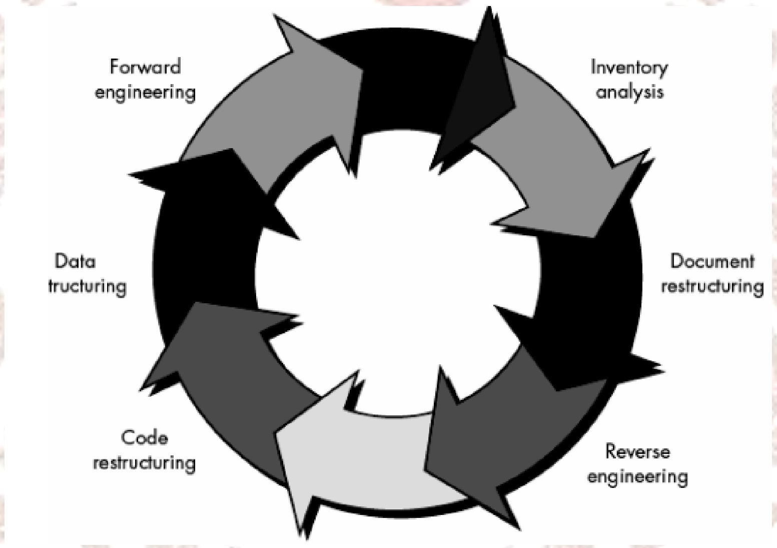


Figure 7.1: A software reengineering process model

i. ***Inventory analysis:*** Every software organization should have an inventory of all applications. The inventory can be nothing more than a spreadsheet model containing information that provides a detailed description (e.g., size, age, business-critical) of every active application by storing the information according to business critically, longevity, current maintainability, and other locally important criteria, candidate for reengineering work. It is important to note that the inventory should be revisited on a regular cycle. The status of applications (e.g., Business critically) can change as a function of time, and as a result, priorities for reengineering will shift.

ii. ***Document restructuring:*** Weak documentation is the trademark of many legacy systems. But what do we do about it? What are our options?

a) Creating documentation is too time-consuming. If the system works, we'll live with what we have. In some cases, this is the correct approach. It is not possible to re-create documentation for hundreds of computer programs. If a program is relatively static, is coming to the end of its useful life, and is unlikely to undergo significant change.

b) Documentation must be updated, but we have limited resources. We'll use a "document when touched" approach. It may not be necessary to fully re-document an application. Rather, those portions of the system that are currently undergoing change are fully documented. Over time, a collection of useful and relevant documentation will evolve.

c) The system is business critical and must be fully re-documented. Even in this case, an intelligent approach is to pare documentation to an essential minimum. A software of organization must choose the one that is most appropriate for each case.

iii. ***Reverse Engineering:*** Reverse engineering is the process of analysing software with the objective of recovering its design and specification. The software source code is usually available as input to the reverse engineering process. Reverse engineering is different from re-engineering. Reverse engineering's purpose is to derive the design or specification of a system from its source code, while the objective of re-engineering is to produce a new, more maintainable system.

iv. ***Code restructuring:*** The common type of reengineering is code restructuring. Some legacy systems have a relatively solid program architecture, but individual modules were coded in a way that makes them difficult to understand, test, and maintain, in such cases, the code within the suspect modules can be restructured. To accomplish this activity, the source code is analysed using are structuring tool, Violations of structured programming constructs are noted, and the code is then restructured. The resultant restructured code is reviewed and tested to ensure that no anomalies have been introduced, and internal code documentation is updated.

v. ***Data structuring:*** A program with weak data architecture will be difficult to adapt and enhance. In fact, for many applications, data architecture has more to do with the long-term viability of a program than the source code itself. Unlike code restructuring, which occurs at a relatively low level of abstraction, data structuring is a full-scale reengineering activity. In most cases, data restructuring begins with a reverse engineering activity. Data objects and attributes are identified, and existing data structures are reviewed for quality. When the data structure is weak (e.g., Flat files are currently implemented, when a relational approach would greatly simplify the process), the data is re-engineered. Because data architecture has a strong influence on program architecture and algorithms that populate it, changes to the data will invariably result in either architectural or code-level changes.

vi. ***Forward Engineering:*** Forward engineering, also called renovation or reclamation, not only recovers design information from existing software but uses this information to alter or reconstitute the existing system to improve its overall quality. In most cases, reengineered software reimplements the function of the existing system and adds new functions and/or improves overall performance.

6. SOFTWARE REFACTORING

Software refactoring refers to the process of making changes to the internal structure or design of a software system while preserving its external behaviour. Configuration management plays a crucial role in supporting software refactoring activities by providing mechanisms for change control, version management, and documentation.

Software refactoring in configuration management involves the following steps:

1. **Analysis:** The software system is analysed to identify areas that can be improved through refactoring. This analysis may involve examining the codebase, design patterns, code smells, performance bottlenecks, or maintainability issues.
2. **Refactoring Identification:** Based on the analysis, specific refactoring techniques or patterns are identified to address the identified issues. Refactoring techniques include methods like extracting methods, renaming variables, removing code duplication, improving code structure, or optimizing algorithms.
3. ***Change Control:*** Configuration management helps in controlling and managing the refactoring changes. This involves tracking and documenting the proposed refactoring modifications, establishing a baseline for the existing system, and ensuring proper version control of the software components involved in the refactoring process.
4. ***Version Management:*** During software refactoring, different versions of the software system may coexist, including the original version and the refactored versions. Configuration management facilitates the management of these versions, ensuring that the changes are properly documented, labelled, and integrated into the overall configuration.
5. ***Integration and Testing:*** The refactored components are integrated into the software system, ensuring that they work seamlessly with the existing components. Configuration management helps in tracking and managing the integration process, including the resolution of any conflicts or issues that arise during the integration phase. Testing is also performed to validate the changes and ensure the overall quality and functionality of the refactored software.
6. ***Documentation:*** Configuration management ensures that proper documentation is maintained throughout the refactoring process. This includes documenting the

refactoring changes made, recording the rationale behind the modifications, and updating any relevant software documentation, such as architectural diagrams, design specifications, and user manuals.

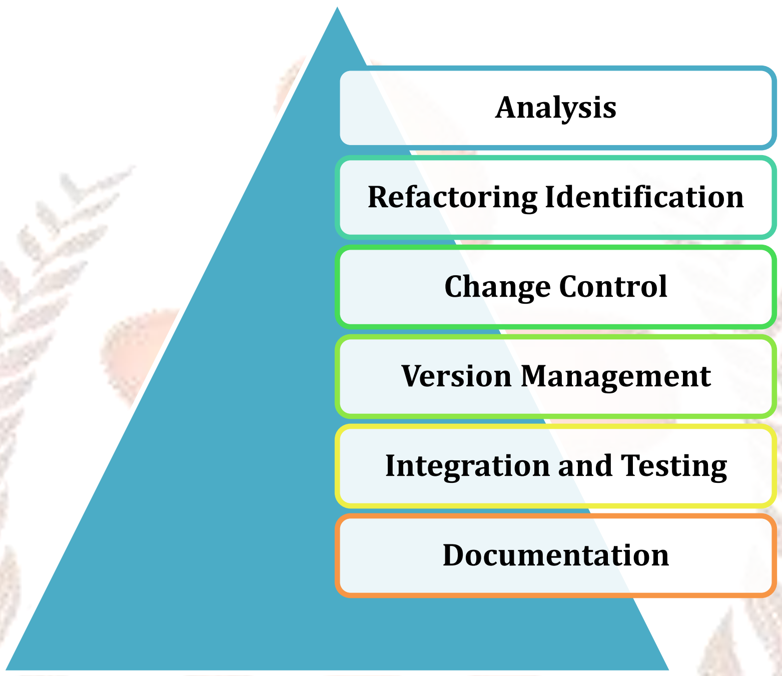


Figure 7.2: Steps Involved in Software Refactoring in Configuration Management

7. SUMMARY

Configuration Management is a critical discipline in software engineering that supports Change Management, Version and Release Management, Software Maintenance, Software Reengineering, and Software Refactoring. It ensures the effective management and control of software components and configurations throughout their lifecycle, enabling efficient software development, maintenance, and evolution.

Configuration management and version and release management are closely related. Version management is concerned with controlling various software component versions, keeping track of their modifications, and keeping a transparent version history. On the other side, release management involves packaging and distributing software releases to end users. Each release of the software is guaranteed to have the most recent versions of those components because of configuration management, which also offers traceability for effective version and release control. The process of modifying a software system to fix bugs, improve functionality, or satisfy new requirements is known as software maintenance. By identifying the components that need adjustments, monitoring the changes performed, and assuring correct integration and testing, configuration management is essential to software maintenance. It makes it easier to organise maintenance tasks effectively and keeps an accurate record of changes for later use.

Software Reengineering is making changes to an existing software system to enhance its performance, maintainability, or other features. Configuration management aids in locating the reengineering-needed components, controlling the changes done, and maintaining the consistency and integrity of the system. The successful completion of the reengineering process is made easier by the documentation and traceability of changes it provides.

Software refactoring focuses on enhancing the software system's internal structure without altering its external behaviour. By identifying the components that need to be refactored, monitoring the changes performed, and guaranteeing appropriate integration, configuration management contributes to software reworking. It makes managing various refactored component versions easier and keeps a detailed record of all reworking efforts.

8. SELF-ASSESSMENT QUESTIONS

1. Reengineering is a __________ activity.
2. Mention any two activities in the software reengineering process model.
3. ____________ is the process of analysing software with the objective of recovering its design and specification.
4. _____________ also called renovation or reclamation.
5. Reengineering is a __________ activity.
6. Mention any two activities in the software reengineering process model.
7. ____________ is the process of analysing software with the objective of recovering its design and specification.
8. _____________ also called renovation or reclamation

9. SELF-ASSESSMENT ANSWERS

1. Rebuilding
2. Inventory analysis and Document restructuring
3. Reverse engineering
4. Forward engineering
5. Change management
6. Set of tools, skills, processes and principles, Acceptance of change by the people
7. Ensure that the reason for the change is clear, Identify the important people who will be involved in the change activities.
8. Mergers or acquisitions, Office relocation

10. TERMINAL QUESTIONS

1. With the help of a diagram explain the software reengineering process model.
2. Describe elements and objectives of change management.
3. Explain some common issues faced by change management.
4. Define Software maintenance and explicate its various tasks.
5. Write a short note on Software refactoring.
6. Elucidate the principal tasks of version and release management.

11. TERMINAL ANSWERS

1. Refer to section 7.5.
2. Refer to section 7.2.1 and 7.2.2
3. Refer to section 7.2.3
4. Refer to section 7.4.
5. Refer to section 7.6.
6. Refer to section 7.3.