Laboratory Practical Report

of

Software Engineering and Project Management

(ICT ED 457)

Submitted To

TRIBHUVAN UNIVERSITY

In Partial Fulfillment of the Requirements of the course

B.Ed. ICTE 5th Semester

Submitted By

Sanam Tamang

Symbol No.: 76214020

T.U. Regd. No.: 9-2-214-54-2019

Under the guidance of

Mr. Atul Bhattarai

Lecturer

Sukuna Multiple Campus

SUKUNA MULTIPLE CAMPUS

Sundarharaincha-12, Morang, Nepal

2080

CERTIFICATE

This is to certify that the Laboratory Practical Report

of

Software Engineering and Project Management

(ICT ED 457)

In Partial Fulfillment of the Requirements of the course

B.Ed. ICTE 5th Semester

Submitted By

Sanam Tamang

Symbol No.: 76214020

T.U. Regd. No.: 9-2-214-54-2019

is a bonafide record of experiments carried out by him/her under the guidance of

_____

Mr. Atul Bhattarai

Lecturer

Sukuna Multiple Campus

Sundarharaincha-12, Morang

(Internal Examiner)

Submitted for the Final Examination on: 2080/02/

_____

Lecturer

(External Examiner)

Contents

1. Explain waterfall model with neat diagram.

The waterfall model is a sequential software development process that follows a linear and phased approach. It consists of distinct phases that are completed one after another, and each phase has well-defined inputs, outputs, and deliverables. The typical phases in the waterfall model are requirements gathering, system design, implementation, testing, deployment, and maintenance.

Here's a brief overview of each phase:

- Requirements Gathering: In this phase, the project requirements are collected and documented. The focus is on understanding the client's needs and defining the system's functional and non-functional requirements.

- System Design: Based on the requirements, a system design is created, which outlines the architecture, components, and interactions of the software system.

- Implementation: The system design is translated into actual code during this phase. Programmers write code following the design specifications and coding standards.

- Testing: The software is thoroughly tested to identify and fix any defects. Different types of testing, such as unit testing, integration testing, and system testing, are performed to ensure the software meets the specified requirements.

- Deployment: Once the software passes testing, it is deployed to the production environment and made available to the end-users.

- Maintenance: After deployment, the software undergoes maintenance, which includes bug fixes, updates, and enhancements to address issues and improve functionality based on user feedback.

The waterfall model follows a strict linear progression, with each phase dependent on the completion of the previous phase. This model is suitable for projects where the requirements are well-defined and stable, and changes are less likely to occur as it can be challenging to accommodate changes once a phase is completed.

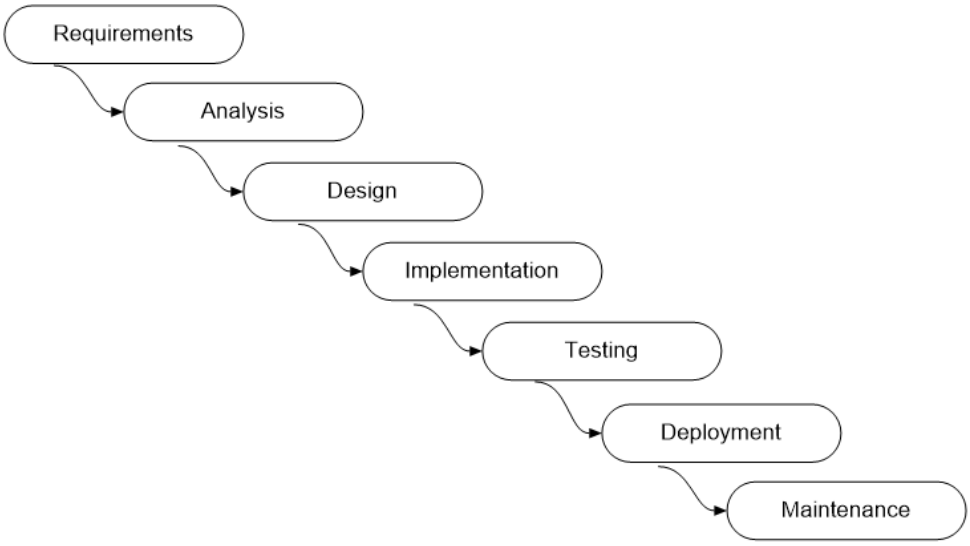

Fig: Waterfall model phases

2. Explain COCOMO model.

COCOMO (Constructive Cost Model) is a software cost estimation model developed by Barry Boehm. It helps in estimating the effort, time, and cost required to develop a software project based on its size, complexity, and other factors. COCOMO is widely used in the industry to estimate project resources and make informed decisions.

COCOMO consists of three different models:

- Basic COCOMO: This model is suitable for estimating the effort and schedule for small to medium-sized projects. It estimates the effort based on the project's estimated lines of code (LOC) and considers factors such as the development team's experience, product complexity, and development environment.

- Intermediate COCOMO: This model includes additional factors such as project attributes, personnel capability, and risk management. It provides a more accurate estimate by considering specific characteristics of the project and the development team.

- Detailed COCOMO: This model provides a more comprehensive estimation by considering a wide range of factors, including cost drivers, development tools, schedule constraints, and more. It takes into account the specific attributes and characteristics of the project to provide a detailed cost and effort estimate. COCOMO models use mathematical formulas and historical

data to estimate the project's effort, schedule, and cost. These estimates help in resource allocation, project planning, and risk assessment. However, it's important to note that COCOMO provides an estimation and may not be 100% accurate, as software development projects can be complex and subject to various uncertainties.

3. Explain coupling and cohesion with examples.

Coupling and cohesion are two fundamental concepts in software engineering that describe the relationships between modules or components within a system.

- Coupling: Coupling refers to the degree of interdependence or connection between modules in a software system. It measures how much one module relies on another. Low coupling is desirable as it indicates loose connections and promotes modularity and flexibility. High

coupling can lead to a lack of modularization, making the system more difficult to maintain and modify.

Example: If two modules are tightly coupled, changes in one module may require modifications in the other module. For instance, if a module relies heavily on global variables or directly accesses the internal implementation of another module, it exhibits high coupling.

- Cohesion: Cohesion refers to the degree of functional relatedness within a single module or component. It measures how well the internal elements of a module work together to achieve a single, well-defined purpose. High cohesion indicates that a module performs a specific and focused task, making it easier to understand, test, and maintain.

Example: A module that handles file input/output operations exhibits high cohesion. All the functions within the module are related to file operations, such as opening, reading, writing, and closing files.

Maintaining low coupling and high cohesion is essential for producing modular, maintainable, and reusable software systems.

4. Explain the types of software maintenance.

Software maintenance involves modifying, enhancing, and fixing issues in software after it has been deployed. There are four main types of software maintenance:

- Corrective Maintenance: Corrective maintenance involves addressing and fixing defects or bugs discovered in the software during its use. This type of maintenance aims to restore the software to its intended functionality and correctness.

- Adaptive Maintenance: Adaptive maintenance involves modifying the software to adapt it to changes in the environment, such as changes in operating systems, hardware configurations, or external dependencies. The purpose is to ensure the software remains compatible and functional in the evolving environment.

- Perfective Maintenance: Perfective maintenance focuses on enhancing the software's performance, efficiency, and maintainability. It involves making improvements to existing features or adding new features to meet evolving user needs and expectations.

- Preventive Maintenance: Preventive maintenance aims to proactively identify and eliminate potential issues or bottlenecks in the software before they cause problems. It involves activities such as code refactoring, performance tuning, and optimization to prevent future issues from occurring.

Each type of maintenance plays a crucial role in ensuring the software remains reliable, efficient, and up-to-date throughout its lifecycle.

5. Explain the ISO 9000 and SEI CMM model.

ISO 9000 is a set of international standards for quality management systems. It provides a framework and guidelines for organizations to establish and maintain effective quality management practices. The ISO 9000 standards focus on processes, documentation, customer satisfaction, and continuous improvement. Compliance with ISO 9000 can demonstrate an organization's commitment to quality and enhance its reputation.

SEI CMM (Capability Maturity Model) is a process improvement framework developed by the Software Engineering Institute (SEI). It defines five levels of maturity that an organization can achieve in its software development processes. The levels are Initial, Repeatable, Defined, Managed, and Optimizing. Each level represents increasing levels of process maturity, effectiveness, and efficiency.

The CMM model provides guidelines and best practices for organizations to improve their software development processes, enhance productivity, and deliver high-quality software. By adopting CMM practices, organizations can systematically assess, plan, and improve their software development capabilities.

Both ISO 9000 and SEI CMM models aim to establish and maintain high-quality standards and practices in software development organizations. While ISO 9000 focuses on overall quality management, the SEI CMM model specifically targets software development processes and their improvement.