Roll no: 2

Student name : Aarya Mayekar

Emailid: aaryamayekar971@gmail.com

Q1 a: Compare FTR and Walkthrough

A Formal Technical Review (FTR) is a Structured software quality control. activity conducted by software engineers to uncover errors in function, logic, or implementation. It is characterized by a formal meeting with a specific agenda, a review leader, and a recorder who produces a formal Summary report. In contrast, a Walkthrough is an informal peer-review process where the author of the work product guides the team through the logic or code to share knowledge and identify bugs early. While FTRS focus on strict adherence to standards, Walkthroughs are often used for early-stage brainstorming.

Q1 c: Explain the LOC

Lines of Code (LOC) is a direct measure of software size used in ank Software engineering

to estimate project effort, cost, and productivity. As a size-oriented metric, it is calculated by counting the total number of lines in a delivered program, typically excluding comments and blank lines. While LOC is easy to calculate and automate, it is highly dependent on the programming language used and can penalize shorter, more efficient code..

It is often used as an input for estimation models like COCOMO to predict development man-months.

Q2 b: Explain the different techniques in white box testing.

Definition: White box testing, also known as structural testing, examines the internal logic and structure of the code.

Control Flow Testing: A strategy using the program's control flow as a model to design test cases.

Basis Path Testing: Allows a designer to derive a logical complexity measure of a procedural design.

Cyclomatic Complexity: A quantitative measure of independent paths in a program. Condition Testing: Exercises the logical conditions contained in a program module.

Data Flow Testing: Selects test paths based on where variables are defined and used.

Loop Testing: Focuses on the validity of loop constructs like simple or nested. loops.

Statement Coverage: Ensures every line of code is executed at least once during testing.

Branch Coverage: Validates every possible outcome from decision points.

Flow Graph Notation: Uses nodes and edges to visualize all executable paths.

Q2 c: Explain steps in version and change control.

Baseline Creation: Establishing a reference point that is formally reviewed and agreed upon.

Change Request: Stakeholders submit formal requests for modifications.

Impact Analysis: Evaluating how the change affects the software and costs.

CRB: An authority reviewing and deciding on the change request.

Check-out: Developers take code from a central repository to avoid simultaneous edits.

Modification: Implementing the approved change in a local environment.

Quality Audit: Testing modified code to ensure it meets requirements.

Check-in: Returning the modified version to the repository with a new version number.

Q3 b: Explain cohesion and Coupling. Explain different types with detailed example.

Core Principle: Design aims for High Cohesion and Low Coupling.

Cohesion Definition: Strength of relationship between elements inside a module.

Functional Cohesion: Module performs exactly one task.

Sequential Cohesion: Output of one process is

the input to another.

Temporal Cohesion: Elements are grouped

because they are performed at the same time.

Coupling Definition: Degree of

interdependence between separate modules.

Data Coupling: Modules communicate by

passing simple data pieces.

Q3 c: Explain the Spiral model of software development.

Definition: An evolutionary model combining

prototyping and waterfall aspects.

Risk-Driven: Uses risk analysis to guide

development at every stage.

Quadrant I (Planning): Determination of

objectives and constraints.

Quadrant 2 (Risk Analysis): Resolution of

technical risks through prototyping.

Quadrant 3 (Engineering): Development of the

product, including coding and testing.

Quadrant 4 (Evalvation): Customer assessment

and planning for the next iteration.

Cumulative Cost: Represented by the distance

from the center as the spiral progresses.

Iterative Nature: Project moves through quadrants in multiple loops.

Flexibility: Allows refinements to requirements at any stage.

Suitability: Best for complex, high-risk projects.

Q4 b: Explain software Re-engineering in detail.

Definition: Analyzing and altering a system to reconstitute it in a new form.

Inventory Analysis: Assessing the portfolio to identify candidates for re-engineering.

Document Restructuring: Updating legacy documentation to reflect current state.

Reverse Engineering: Analyzing a program to identify components and abstractions.

Code Restructuring: Modifying source code for readability without changing functionality.

Data Restructuring: Redesigning data Structures for performance or security.

Forward Engineering: Using reverse-

engineered knowledge to build a new technology.

BPR: Aligning software Software with new organizational goals.

Benefits: Improves maintainability and system reliability.

Cost-Benefit Ratio: Chosen when cheaper than building from scratch.

Q4 c: What are the different types of maintenance?

Software Maintenance: Modifying a system after delivery to fix faults or improve performance.

Corrective Maintenance: Reactive modification to fix discovered bugs.

Adaptive Maintenance: Modifying software to keep it usable in a changed environment.

Perfective Maintenance: Enhancing software by adding features or refining the UI.

Preventive Maintenance: Modifying software to fix latent faults before they become effective.

Lifecycle Cost: Maintenance accounts for a

large percentage of total lifecycle cost.

Business Relevance: Ensures the system remains relevant to the business.

Reliability: Improves the overall reliability of the software.

Maintainability: Increases the ease with which software can be maintained.

Post-delivery: These activities strictly occur after the software has been delivered.