Roll no: 5

Student name: Amit Patil

Email id : sailatke333@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 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 a: Explain Risk and its types? Explain the RMMM plan

Risk Definition: Potential project problem that

might occur, affecting the overall project

schedule, budget, or quality.

Project Risks: Threats like schedule delays,

budget overruns, or significant resource loss

affecting the project plan.

Technical Risks: Hurdles like unproven tech,

complex algorithms, or specific hardware

limitations during the implementation phase.

Business Risks: Market viability concerns, competitor products, or a sudden shift in

consumer needs and demands.

Known/Predictable: Evalvated project plan

risks versus those extrapolated from past

project experience and historical data.

Mitigation (R): Proactive reduction steps to

decrease the overall probability or potential

impact of a risk.

Monitoring (M): Tracking risk likelihood and

checking if the identified proactive mitigation is

actually working effectively.

Management (M): Steps for real problems

focusing on essential contingency planning to

resolve issues as they occur.

RMMM Plan: Documents risks and owners

including the specific probability and the

potential impact of each risk.

Goal: Systematic preparation for issues to

ensure overall project stability by anticipating

what could go wrong.

Q2 b: Explain the different techniques in white box testing

White box testing: Structural code analysis of

the internal logic.

Control Flow: Testing decision points for every

program path.

Basis Path: Execution path guidance for the

test designer.

Cyclomatic Complexity: Complexity metric

defining the number of paths.

Condition Testing: Logic condition exercise for

true and false.

Data Flow: Variable based paths for definitions

and uses.

Loop Testing: Loop validity focusing on the loop

constructs.

Statement Coverage: Single line execution for

every source line.

Branch Coverage: Outcome validation for

every possible decision branch.

Flow Graph: Path visualization using nodes and

connecting edges.

Q3b: Explain cohesion and Coupling

Principle: Aim for High Cohesion and Low

Coupling for modularity.

Cohesion: Internal strength of a module for

functional unity.

Functional: Single task module performing

exactly one function.

Sequential: Process output used as input for

next process.

Temporal: Grouping by time for initialization

routine tasks.

Coupling: Interdependence between modules

that are separate software.

Data: Passing parameters between modules for

communication exchange.

Control: Using flags to direct logic of another

module.

Common: Sharing global data area causing

potential integrity issues.

Content: Modifying other module data which is

the worst form.

Q3 c: Explain the Spiral model of software development.

Definition: Evolutionary model for software

development process iterations.

Risk-Driven: Uses risk analysis at every stage

of development.

Planning: Objective determination of project

alternatives and constraints.

Risk Analysis: Resolving technical risks

through prototyping and simulation.

Engineering: Development and testing including

coding and integration.

Evalvation: Assessment of work by customer

for next circuit.

Cost: Shown by spiral distance from the

starting center.

Iterative: Project moves in loops until the

system is complete.

Flexibility: Allows refinements to requirements

at any project stage.

Suitability: Complex projects where

requirements may evolve over time.

Q4 a: Explain the general format of SRS for Hospital Management system

Introduction: Defines the purpose and

objectives of HMS.

Description: Outlines user roles like Doctors

and Patients.

Registration: Manages admission details and

electronic medical records.

Appointments: Logic for scheduling

consultations and doctor-patient tracking.

Billing: Requirements for payment

requirements and insurance processing.

Security: Ensuring data privacy via RBAC

access control.

Availability: Guarantees constant 24/7 access

for emergency services.

Interface: Managing hardware interaction and

software interface requirements.

Database: Secure patient record storage and

data schema.

Appendices: Includes diagrams like Level O

DFD workflows.

Q4 b: Explain software Re-engineering in detail.

Definition: Analyzing and altering a system to improve maintainability.

Inventory: Assessing portfolio to identify re-engineering candidates effectively.

Document: Updating documentation for legacy systems to reflect logic.

Reverse Engineering: Identifying components and their interrelationships for abstractions.

Code Restructuring: Making code readable and efficient without changing functionality.

Data Restructuring: Redesigning schemas to improve performance and security.

Forward Engineering: Building with new tech using reverse engineering knowledge.

BPR: Aligning with goals through business process re-engineering.

Benefits: System reliability and overall improvement in software quality.

Cost: Cheaper than starting over from a new system.