**SOFTWARE ENGINEERING**

**QUESTION TWO**

**Discuss the following software testing techniques (walkthroughs, reviews and inspections, dynamic testing, Traceability matrices. Debugging environments)**

**Walkthroughs.**

In the walkthrough, the code or document is read by the author, and others who are present in the meeting can note down the important points or can write notes on the defects and can give suggestions about them. The walkthrough is an informal way of testing; no formal authority is being involved in this testing.

**Reviews and Inspections.**

These are considered as the most formal type of review. These are usually conducted after product accomplishes predefined exit criteria that is a particular requirement is implemented.

In a review also known as technical review, a work product is especially examined for defects by several individuals other than the person who actually produced it. In this, work product is defined as an essential and important deliverable that is created during requirements, design, coding, or a testing phase of software development.

**Dynamic Testing.**

Dynamic testing is a type of software testing that involves executing the software and evaluating its behavior during runtime. It is also known as functional testing, as it focuses on testing the software’s functionality and how it behaves under different inputs and conditions.

The software is run and tested against a set of predefined test cases. These test cases are designed to cover a range of inputs and use cases, and they are used to check the software’s behavior and output in response to different inputs. This can include testing the software’s user interface, functional requirements, and performance.

**Traceability Matrices.**

Used to trace the requirements to the tests that are needed to verify whether the requirements are fulfilled.

Traceability matrices link different elements of the software development process, such as requirements, design, and test cases.

They ensure that every requirement has associated design and test coverage, helping maintain consistency and completeness throughout the software development lifecycle.

**Debugging Environment.**

Debugging is the process of fixing a bug in the software. It can be defined as identifying, analyzing, and removing errors.

This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

**Discuss Software Quality Evaluation – Problems, Software standards, Certification, Software Tools support for Systems Engineering.**

Software quality evaluation determines a software application or system's quality, usability, and effectiveness. It is an important step in the software development process because it determines whether the software meets the required standards and specifications and is fit for its intended purpose.

**Problems.**

* Subjectivity: Evaluating software quality is often subjective and influenced by individual opinions. Different stakeholders may have varying perceptions of quality.
* Changing Requirements. Software quality evaluation can be challenging when requirements are ambiguous or continuously changing. Assessing whether the software meets evolving requirements can be complex.
* Resource Constraints. Limited time, budget, and resources can restrict the thoroughness of quality evaluation. Inadequate testing and insufficient review processes may result from these constraints.
* Measuring Non-Functional Attributes. Non-functional attributes such as performance, security, and usability can be difficult to measure objectively, making it challenging to evaluate software quality in these areas.

**Software Standards.**

* ISO 25010. This standard defines a comprehensive framework for software product quality. It categorizes quality characteristics into eight attributes, including functionality, reliability, and efficiency.
* ISO 9001. ISO 9001 is a quality management system standard that can be applied to software development. It focuses on process quality and continuous improvement.
* CMMI (Capability Maturity Model Integration). CMMI is a process improvement framework that helps organizations optimize their processes for enhanced software quality and productivity.

**Certification.**

* Credibility. A certified software product is considered more reliable, and potential users or customers have confidence in its quality.
* Market Access. In some sectors, certification may be required to access certain markets or contracts. For example, medical software may require certification to meet regulatory requirements.
* Competitive Advantage. Certification can provide a competitive advantage by demonstrating a commitment to high-quality software development.

**Software Tools Support for Systems Engineering.**

* Requirements Management Tools. Tools like IBM DOORS or Jama Connect help in capturing, managing, and tracing system requirements, ensuring that the system meets the desired functionality and performance.
* Modeling and Simulation Tools. Software like SysML modeling tools (e.g., MagicDraw, Enterprise Architect) enables the creation of system architecture and behavior models, allowing for analysis and simulation.
* Configuration Management Tools. Tools like Git and SVN help manage versions of system artifacts and control changes to ensure consistency and traceability.
* Project Management Tools. Tools like Jira, Trello, and Microsoft Project assist in planning, tracking, and managing the progress of systems engineering projects.
* Collaboration Tools. Collaboration tools like Confluence or SharePoint enable teams to communicate, share documents, and collaborate on system engineering tasks.
* Testing and Verification Tools. Tools for system verification and validation, such as MATLAB/Simulink, help assess the system's performance and functionality.

**Discuss CASE tools as used in Web engineering techniques and process, standards and guidelines.**

**Web Engineering Techniques and Process.**

**Requirements Gathering.**

CASE tools support gathering, documenting, and managing web application requirements. They provide features to create requirement models, use cases, and user stories.

**Design and Modeling.**

CASE tools aid in designing the architecture and layout of web applications. They offer visual tools for creating wireframes, UI/UX designs, and site structures.

UML (Unified Modeling Language) tools within CASE assist in visualizing system components, interactions, and data flow.

**Development and Coding.**

Integrated Development Environments (IDEs) within CASE tools provide features for coding web applications. They offer syntax highlighting, debugging, version control, and code analysis.

**Testing and Debugging.**

Some CASE tools offer functionalities for generating test cases, running tests, and debugging web applications. They aid in identifying and resolving issues in the code.

**Deployment and Maintenance.**

CASE tools may support deployment processes by integrating with deployment platforms or providing features for release management. They can also assist in maintaining and updating web applications.

**Standards and Guidelines.**

**Adherence to Web Development Standards.**

CASE tools often come with templates and libraries that comply with web development standards like W3C guidelines, ensuring the application's adherence to HTML, CSS, and accessibility standards.

**Usability and User Experience (UX).**

Many CASE tools offer design modules specifically focused on creating intuitive and user-friendly interfaces, following UX best practices.

**Security Standards.**

Some CASE tools integrate security features and modules that assist in applying best practices for securing web applications, preventing vulnerabilities like SQL injection, cross-site scripting, etc.

**Performance Optimization.**

CASE tools might incorporate modules for analyzing and optimizing web application performance, ensuring adherence to performance standards for faster loading times and responsiveness.

**Discuss Process Improvement (PI):- Quality and process standards and guidelines.**

Software Process Improvement (SPI) methodology is a sequence of tasks, tools, and techniques to plan and implement improvement activities to achieve specific goals such as increasing development speed, achieving higher product quality or reducing costs.

**Quality Standards:**

* Consistency: Quality standards ensure consistency in product or service delivery. They define best practices that need to be followed consistently across the organization.
* Customer Satisfaction: Meeting quality standards enhances customer satisfaction by delivering products or services that meet or exceed their expectations.
* Compliance: Quality standards often include regulatory and industry-specific requirements, ensuring compliance with legal and industry norms.
* Benchmarking: Quality standards provide a benchmark against which organizations can measure their processes and products, facilitating continuous improvement.

**Process Standards and Guidelines:**

* Best Practices: Process standards incorporate industry-recognized best practices, such as the use of well-established software development methodologies like Agile, Scrum, or Waterfall.
* Documentation: Guidelines often mandate the documentation of processes and procedures, ensuring that they are well-defined and understood by all stakeholders.
* Efficiency: Process standards emphasize process efficiency and effectiveness, driving organizations to optimize their workflows for better results.
* Continuous Improvement: Many process standards promote a culture of continuous improvement, where organizations regularly assess their processes, identify areas for enhancement, and implement changes.