**Omphile Hermina Segone**

[omphilehsegone@gmail.com](mailto:omphilehsegone@gmail.com)

+27 63 882 4540

https://github.com/Powerlearnproject/se-assignment-2-introduction-to-software-engineering-Omphile9926

**What is software engineering**?

The methodical application of engineering techniques to the creation, management, and upkeep of software is known as software engineering. It entails using an organized and disciplined approach to design, develop, test, and manage software systems to guarantee that they satisfy quality, reliability, and performance standards.

**How does software engineering differ from traditional programming?**

Software engineering emphasizes the application of approaches like Agile, Waterfall, or DevOps and takes a methodical and disciplined approach to software development. These approaches specify a sequence of actions or stages, including requirements analysis, design, implementation, testing, deployment, and maintenance, that direct the development process. This organized technique may be absent from traditional programming, resulting in ad hoc development practices.

**Explain the various phases of the Software Development Life Cycle:**

**The steps of the Software Development Life Cycle are as follows:**

* Requirement analysis: Compiling and outlining the software's needs.
* Design: Organizing the software's architecture and structure.
* Code is written using the design as a guide throughout implementation.
* Testing: Making sure the program satisfies the requirements by validating and verifying it.
* Software is released for usage through deployment.
* Updating and improving the program during its lifecycle is called maintenance.

**Compare and contrast the Agile and Waterfall models of software development**.

The agile methodology places a strong emphasis on cross-functional teams, flexibility, continuous delivery, customer collaboration, iterative and incremental development, and little documentation. It centers on brief intervals known as sprints, which enable teams to react rapidly to modifications in specifications and client demands. Agile prioritizes functional software over extensive documentation.

**Preferred Scenarios for Agile:**

* Agile is best suited for tasks with an ambiguous scope, client participation, small teams, startups, frequent releases, and short time-to-market.
* T he sequential software development methodology known as the Waterfall Model places a strong emphasis on milestones, predictability, and documentation.
* It is less flexible, though, and it could result in problems being found later in the process. Since testing is usually completed at the very end, silos between teams or departments may result. All in all, it's a difficult and drawn-out process.

**Preferred Scenarios for Waterfall:**

* Large-scale, regulated, stable, and predictable projects with well-defined requirements, logical steps, copious documentation, and a defined scope are examples of waterfall situations.

**What is requirement Engineering?**

A methodical and disciplined technique to gathering, recording, evaluating, validating, and overseeing the requirements for a software system is known as requirements engineering. This crucial stage of the software development lifecycle (SDLC) involves identifying the needs and expectations of stakeholders and turning them into a set of precise requirements that form the basis of the development procedure.

**Process of Requirements Engineering:**

During the requirements phase, different techniques are used to collect needs from stakeholders. After that, an analysis is conducted to make sure these requirements are feasible, thorough, consistent, and clear. A formal specification document contains the requirements. The requirements are checked using validation techniques to make sure they appropriately reflect the needs of the stakeholders. Monitoring changes made during the software development lifecycle, setting priorities, handling dependencies, and preventing scope creep are all part of requirements management.

**Importance of Requirements Engineering:**

In software development, requirements engineering is an essential process that guarantees regulatory compliance in sectors like healthcare, finance, and aerospace while also reducing development costs and risks, fostering better communication and teamwork, and enhancing quality and customer satisfaction. It guarantees a strong basis for the development process and reduces project failure, rework, and delays.

**The concept of modularity in software design.**  
The process of dividing a software system into more manageable, independent components or modules, each of which carries out a particular task or function, is known as modularity software design. With clearly defined interfaces that enable interaction and communication between them, these modules are made to be independent from one another.

**How modularity improves maintainability and scalability of software systems:**

**Maintainability**:

Code reuse, faster testing and debugging, and maintainability are just a few advantages of a modular structure. It makes the codebase easier to comprehend and manage by enabling changes to be made to one module without affecting other sections. Additionally, this enhances program consistency and makes maintenance easier.

**Scalability**:

Modular architecture allows for resource scalability, concurrent development, and flexibility in adding functionality. It makes it simple to add new modules or extensions, which enables software to change to meet evolving needs without requiring a lot of reworks. It also makes it possible to allocate resources for modules effectively.

**Testing in Software Engineering:**

* Unit testing is the process of testing separate modules or components separately.
* Integration testing is the process of examining how several modules or components work together.
* System testing is the process of examining the system.
* Testing the software to make sure it satisfies the acceptance standards established by the stakeholders is known as acceptance testing.
* To ensure the quality and dependability of the program, testing is essential during the software development process to find and correct bugs early on.

**What are version control systems?**

They are computer programs designed to handle alterations made to files, documents, and code over time. They offer a central location where developers may work together, save, and monitor a software project's development.

**why version control systems are important in software development:**

Developers may review and go back to previous versions of a file thanks to version control systems (VCS), which offer a thorough history of file modifications. By enabling several developers to collaborate on the same codebase at once and providing backup and disaster recovery, they promote teamwork. Developers can also start forks. Furthermore, VCS facilitates quality assurance and code reviews, guaranteeing consistency and early issue detection.

**Examples of popular version control systems and their features.**

**Git**  
system of distributed version control.  
capability for branching and merging.  
swift and lightweight.  
  
**Subversion (SVN):**centralized system for version control.  
conventional architecture of clients and servers.  
support for versioning folders and atomic commits.  
   
**Erratic:**  
system of distributed version control.  
Simple to operate and lightweight.  
Support for branching and merging is integrated.   
  
**Helix Core Perforce:**centralized system for version control.  
both enterprise-grade and scalable.  
high-speed file retrieval and storage.

**The role of a software project manager**

Supervising the development, implementation, and completion of software projects is a critical responsibility of the software project manager. They are in charge of making sure that projects are finished on schedule, under budget, and in compliance with specifications.

**Key Responsibilities:**

* Project managers for software are in charge of establishing the goals, parameters, schedules, and resource needs of their projects. They draft project plans that include duties, deadlines, dependencies, and completion dates.
* To make sure that project operations are properly resourced and carried out effectively, project managers assign resources, such as personnel, funds, and equipment.
* In order to reduce the influence of possible risks on a project's success, project managers must identify, evaluate, and mitigate those risks. They must then develop risk management plans.

**Key challenges:**

* Complex technical issues, such merging disparate systems, increasing performance, or guaranteeing security, are frequently encountered in software projects.
* Changes in scope require careful management by project managers in order to avoid delays, overspending, and problems with quality.

**What is software maintenance?**

The practice of making changes and updates to a software product after it has been released in order to maintain its efficacy, efficiency, and utility throughout its lifecycle is known as software maintenance.

**Types of maintenance activities:**

* Corrective maintenance is the process of identifying and resolving issues or flaws in software that are found after it has been implemented.
* Adaptive maintenance refers to the process of changing software to account for modifications to its working environment, such as updates to hardware or changes in user preferences.
* Emergency maintenance entails taking care of urgently needed critical issues in order to avoid system failure or major operational disruptions.
* Preventive maintenance aims to proactively identify and address potential issues before they manifest as problems this includes activities such as code refactoring.
* Emergency maintenance entails taking care of urgently needed critical issues in order to avoid system failure or major operational disruptions.

**Why is maintenance an essential part of the software lifecycle?**

In the software lifecycle, maintenance is essential for resolving bugs, improving features, maximizing performance, mitigating security risks, and guaranteeing long-term viability. It aids in the detection, correction, and defence against cyberattacks of defects by developers.

**What are some ethical issues that software engineers might face?**

The job of software engineers presents a variety of ethical conundrums, such as security issues, fairness and bias, accountability and transparency, societal effect, and whistleblowing. They must take into account potentially biased algorithms and sensitive user data. They also must accept accountability for software errors and protect intellectual property rights. It could be difficult for software engineers to report unethical or illegal activity, which could have an effect on their employment.

**How can software engineers ensure they adhere to ethical standards in their work?**

* Staying current with laws, ethical standards, and professional behaviour regulations that apply to their sector of work.
* creating a framework for moral decision-making that takes into account the possible effects of their choices on society, the environment, and stakeholders.
* putting user privacy first and getting consent after being informed while gathering, keeping, or utilizing personal data.
* putting user privacy first and getting consent after being informed while gathering, keeping, or utilizing personal data.
* making an effort to create impartial, inclusive, and fair algorithms and systems, as well as taking precautions against potential biases.