SOFTWARE ENGINEERING ASSIGNMENT

INTRODUCTION

Software engineering is the branch of computer science that deals with the design, development, testing, and maintenance of software applications. Software engineers apply engineering principles and knowledge of programming languages to build software solutions for end users.

**The importance of software engineering.**

1. Reduces Complexity

Dealing with big Software is very complicated and challenging. Thus, to reduce the complications of projects, software engineering has great solutions. It simplifies complex problems and solves those issues one by one.

1. Handling Big Projects

Big projects need lots of patience, planning, and management, which you never get from any company. The company will invest its resources; therefore, it should be completed within the deadline. It is only possible if the company uses software engineering to deal with big projects without problems.

1. To Minimize Software Costs

Software engineers are paid highly as Software needs a lot of hard work and workforce development. These are developed with the help of a large number of codes. But programmers in software engineering project all things and reduce the things which are not needed. As a result of the production of Software, costs become less and more affordable for Software that does not use this method.

1. To Decrease Time

If things are not made according to the procedures, it becomes a huge loss of time. Accordingly, complex Software must run much code to get definitive running code. So, it takes lots of time if not handled properly. And if you follow the prescribed software engineering methods, it will save your precious time by decreasing it.

1. Effectiveness

Making standards decides the effectiveness of things. Therefore, a company always targets the software standard to make it more effective. And Software becomes more effective only with the help of software engineering.

1. Reliable Software

The Software will be reliable if software engineering, testing, and maintenance are given. As a software developer, you must ensure that the Software is secure and will work for the period or subscription you have agreed upon.

**Milestones of software engineering**

Software engineering is a relatively young discipline, but its roots can be traced back to the early days of computing. The field has evolved dramatically over time, with new challenges and opportunities emerging at each stage.

Early Days (1940s-1960s)

The first computers were massive and expensive, and programming them was a complex and labor-intensive task. Early programmers often had to write machine code, which is a low-level language that directly controls the computer's hardware.

Key milestones:

1948: Grace Hopper develops the first compiler, which translates human-readable code into machine code.

1957: FORTRAN, the first high-level programming language, is released.

1960s: The term “software engineering” is coined.

Challenges:

* Programs were difficult to write and debug.
* There were few tools and techniques available to help developers.
* The field lacked a formal body of knowledge.

**Software development life cycle**

This is the process of creating a detailed plan to guide the development of the product and then breaking down the development process into smaller modules that can be assigned, completed, and measured to make the whole thing more manageable.

The 7 Phases of the Software Development Life Cycle

1. **Planning & Analysis**

The first phase of the SDLC is the project planning stage where you are gathering business requirements from your client or stakeholders. This phase is when you evaluate the feasibility of creating the product, revenue potential, the cost of production, the needs of the end-users.

Once it is decided that the software project is in line with business and stakeholder goals, feasible to create, and addresses user needs, then you can move on to the next phase.

1. **Define Requirements**

This phase is critical for converting the information gathered during the planning and analysis phase into clear requirements for the development team. This process guides the development of several important documents: a software requirement specification (SRS) or product specification, a Use Case document, and a Requirement Traceability Matrix document.

1. **Design**

The design phase is where you put pen to paper. The original plan and vision are elaborated into a software design document (SDD) that includes the system design, programming language, templates, platform to use, and application security measures. This is also where you can flowchart how the software responds to user actions.

1. **Development**

The actual development phase is where the development team members divide the project into software modules and turn the software requirement into code that makes the product.

This SDLC phase can take quite a lot of time and specialized development tools. It’s important to have a set timeline and milestones so the software developers understand the expectations and you can keep track of the progress in this stage.

1. **Testing**

Before getting the software product out the door to the production environment, it’s important to have your quality assurance team perform validation testing to make sure it is functioning properly and does what it’s meant to do. The testing process can also help hash out any major user experience issues and security issues.

The types of testing to do in this phase:

* Performance testing: Assesses the software’s speed and scalability under different conditions
* Functional testing: Verifies that the software meets the requirements
* Security testing: Identifies potential vulnerabilities and weaknesses
* Unit-testing: Tests individual units or components of the software
* Usability testing: Evaluates the software’s user interface and overall user experience
* Acceptance testing: Also termed end-user testing, beta testing, application testing, or field testing, this is the final testing stage to test if the software product delivers on what it promises

1. **Deployment**

During the deployment phase, your final product is delivered to your intended user. You can automate this process and schedule your deployment depending on the type. For example, if you are only deploying a feature update, you can do so with a small number of users (canary release). If you are creating brand-new software, you can learn more about the different stages of the software release life cycle (SRLC).

1. **Maintenance**

The maintenance phase is the final stage of the SDLC if you’re following the waterfall structure of the software development process. However, the industry is moving towards a more agile software development approach where maintenance is only a stage for further improvement.

In the maintenance stage, users may find bugs and errors that were missed in the earlier testing phase. These bugs need to be fixed for better user experience and retention. In some cases, these can lead to going back to the first step of the software development life cycle.

**The difference between waterfall model and agile methodology**

1.Definition; Agile model follows the incremental approach, where each incremental part is developed through iteration after every time box while waterfall model follows a sequential design process.

2.Progress; In the agile model, the measurement of progress is in terms of developed and delivered functionalities while in the waterfall model, generally the measurement of success is in terms of completed and reviewed artifacts.

3. Nature ; Agile model is flexible as there is a possibility of changing the requirements even after starting the development process. On the other hand, the waterfall model is rigid as it does not allow to modify the requirements once the development process starts.

4. Customer interaction; In Agile model, there is a high customer interaction. It is because, after every iteration, an incremental version is deployed to the customer while customer interaction in waterfall model is very less. It is because, in a waterfall model, the product is delivered to the customer after overall development.

5. Team size; It has a small team size. As smaller is the team, the fewer people work on it so that they can move faster while in the waterfall model, the team may consist more members.

6. Suitability; Agile model is not a suitable model for small projects. The expenses of developing the small projects using agile is more than compared to other models while waterfall model works well in smaller size projects where requirements are easily understandable. But waterfall model is not suitable for developing the large projects.

7. Test plan; In agile, the test plan is reviewed after each sprint while in waterfall model test plan is reviewed after complete development.

8. Testing ; In agile testing team can take part in the requirements change phase without problems while in waterfall model it is difficult for the testing team to initiate any change in needs.

The waterfall model is best suited for projects with well-defined requirements. For example, developing software for a regulatory body where all the requirements are fixed upfront due to strict compliance needs. Whereas, agile methodology is ideal for projects with uncertain or rapid changing requirements. For example, a start up developing a new mobile app where the market needs and user feedback are constantly evolving.

**The roles and responsibilities of a project manager**

**Monitoring and reviewing:** Project monitoring is a continuous process that lasts the whole time a product is being developed, during which the project manager compares actual progress and cost reports with anticipated reports as soon as possible. While most firms have a formal system in place to track progress, qualified project managers may still gain a good understanding of the project’s development by simply talking with participants.

**Documenting project report:** The

Project manager prepares the documentation of the project for future purposes. The reports contain detailed features of the product and various techniques. These reports help to maintain and enhance the quality of the project in the future.

**Reporting:** Reporting project status to the customer and his or her organization is the responsibility of the project manager. Additionally, they could be required to prepare brief, well-organized pieces that summarize key details from in-depth studies.

**Risk Management;** the project manager should identify the unanticipated risks that may occur during project development risk, analyze the damage that might cause these risks, and take a risk reduction plan to cope with these risks.

**Miscellaneous Plans;** this includes making several other plans such as quality assurance plans, configuration management plans, etc.

**Lead the team:** The project manager must be a good leader who makes a team of different members of various skills and can complete their individual tasks.

**Motivate the team-member:** One of the key roles of a software project manager is to encourage team members to work properly for the successful completion of the project.

**Roles and responsibilities of a software developer**

The software developer roles and responsibilities include creating and constructing computer programs that run desktop computers, mobile gadgets, and even automobiles. In addition to identifying customer demands, they develop new applications for different industries and make alterations based on user feedback.

**Roles and responsibilities of a quality assurance engineer**

The quality assurance engineers spend most of their time assessing the software for bugs, suggesting changes and ensuring it meets quality standards. They prepare test cases to check the software and document their results. Some of the roles include;

* Create standards and procedures for developers to follow
* Execute both manual and automated test scripts
* Code and implement automated tests
* Communicate with stakeholders to understand and clarify software requirements
* Analyse test results to predict user behaviour, identify bugs and suggest solutions to minimise problems
* Work closely with product development and testing teams to ensure timely delivery of the project
* Keep track of quality metrics
* Verify that the final product meets the requirement
* Analyse product features and suggest changes to enhance them to make them more efficient and user friendly
* Stay up-to-date with the latest industry developments, standards and regulations

**The importance of IDEs and version control system in software development process**

Integrated Development Environments (IDEs) and Version Control Systems (VCS) are critical tools in the software development process. They each play a unique role In enhancing productivity, ensuring code quality, and enabling collaboration among developers. Here’s why they are important:

Integrated Development Environments (IDEs):

1. Code Writing and Editing;

Importance: IDEs provide advanced code editors that offer features like syntax highlighting, auto-completion, and code formatting. These features help developers write code faster and with fewer errors.

2. Debugging Tools ;

Importance: IDEs come with built-in debugging tools that allow developers to set breakpoints, inspect variables, and step through code execution line by line.

Version Control Systems (VCS):

1. Source Code Management

Importance; VCS tools like Git track changes to the source code over time, allowing developers to manage and access different versions of their code.

2. Collaboration and Teamwork:

Importance: VCS allows multiple developers to work on the same project simultaneously without overwriting each other’s changes.

**Challenges faced by software engineers**

Software engineers face a variety of challenges that can impact productivity, code quality, and project success. Here are some common challenges along with strategies to overcome them:

1. Managing Changing Requirements

Challenge: Requirements often change during a project, leading to scope creep, rework, and delays. This is especially challenging in long-term projects where initial requirements may no longer be relevant.

Strategies to Overcome; Adopt Agile Methodologies because it practices like regular sprints and iterative development allow teams to accommodate changes incrementally rather than being overwhelmed by them all at once.

2. Ensuring Code Quality

Challenge: Writing high-quality, maintainable code is difficult, especially under tight deadlines. Poor code quality can lead to bugs, technical debt, and increased maintenance costs.

Strategies to Overcome:

Code Reviews: Regular peer reviews help catch issues early, promote knowledge sharing, and ensure adherence to coding standards.

Automated Testing: Implement unit tests, integration tests, and continuous integration (CI) pipelines to automatically validate code quality with every change.

**The types of testing and their importance in software quality assurance**

1. Unit Testing; This involves testing individual units or components of a software application. A unit typically refers to a single function, method, or class.

Unit testing is important because it ensures the functionality and correctness of individual components, leading to better code quality and easier maintenance.

1. Integration Testing; This involves testing the interaction between integrated units or components of the software to ensure they work together as expected.

Importance; Verifies that integrated components work together correctly, preventing issues that could arise from component interactions.

3. System Testing; System testing involves testing the complete and integrated software system as a whole. It validates the system against the functional and non-functional requirements.

Importance; Confirms that the complete system meets all specified requirements, ensuring it functions correctly in its entirety.

4. Acceptance Testing; Acceptance testing is the final phase of testing before the software is delivered to the customer. It is performed to ensure that the software meets the business requirements and is ready for use.

Importance; Validates that the software meets the business and user requirements, ensuring it is ready for deployment and use.

**Prompt engineering**

Prompt engineering is the art and science of designing and optimizing prompts to guide AI models, particularly Large Language Models(LLMs), towards generating the desired responses. By carefully crafting prompts, you provide the model with context, instructions, and examples that help it understand your intent and respond in a meaningful way.

**The importance of prompt engineering**

**Improved model performance;** Well-crafted prompts lead to more accurate, relevant, and informative outputs from AI models, as they provide clear instructions and context.

**Reduced bias and harmful responses;** By carefully controlling the input and guiding the AI’s focus, prompt engineering helps mitigate bias and minimize the risk of generating inappropriate or offensive content.

**Increased control and predictability;** Prompt engineering empowers you to influence the AI’s behavior and ensure consistent and predictable responses aligned with your desired outcomes.

**Enhanced user experience;** Clear and concise prompts make it easier for users to interact effectively with AI models, leading to more intuitive and satisfying experiences.

**An example of a vague**

Vague prompt:

“I want to complete my PLP training and get a certificate”

Improved prompt:

“I aiming to be finishing an assignment given everyday. Attending every class in the next 3 months by attending 1 hour in every session, 3 hours a day.

The Improved prompt is more effective because:

1. Specificity; It clearly states the desired outcome (losing 10 pounds) and the actions to achieve it (exercise and healthy eating).

2. Measurability: The goal is quantifiable (10 pounds in 3 months), making progress easy to track.

3. Concreteness: The actions are well-defined (30 minutes, 3 times a week, 5 servings) leaving no room for misinterpretation.

4. Achievability: The goals are realistic and attainable, making it more likely to motivate and maintain commitment.