1. **Introduction to Software Engineering**
2. Explain what software engineering is and discuss its importance in the technology industry.

Software engineering refers to a branch in computer science that deals with designing, developing, using and maintaining high quality software systems.

Importance of software engineering in technology industry includes;

* Healthcare management, where systems are developed e.g. Telemedicine platforms and electronic health records. Also, for cybersecurity purposes in order to protect patient data and for automation of routine tasks such as billing and some diagnostic procedures
* Communications industry, where there is development of communication platforms e.g. social media platforms and messaging apps, video conferencing platforms that promote interaction. Also in integration of AI and automated communication tools like virtual assistants and chatbots
* Entertainment industry, for creation of media content (movies, music) and digital art, special effects and animation and various streaming platforms (YouTube, Netflix) and video game development
* Commerce, by developing e-commerce platforms that help in business e.g. amazon, Alibaba; for development of digital payment systems for efficient payment processing e.g. M-pesa, PayPal, google pay etc. and for data analytic tools that allow businesses to analyze and interpret data on sales and market performance

1. Identify and describe at least three key milestones in the evolution of software engineering.

* Structural programming, such as using loops, conditionals, and subroutines, improved code readability and maintainability. This marked the beginning of more disciplined software development practices, setting the stage for future methodologies like modular and object-oriented programming.
* Genesis of the waterfall mode, that was one of the first formalized software development methodologies. It organized the development process into sequential phases: requirements, design, implementation, testing, deployment, and maintenance.
* Rise of agile methodologies, that transformed the way software is developed by promoting continuous feedback and adaptability. It responded to the need for more dynamic and responsive software development processes, especially in a world where customer requirements can change rapidly.

1. List and briefly explain the phases of the Software Development Life Cycle.

* Planning – this involves defining a software and its purpose by pinpointing the destination and plotting the best route
* Requirement analyzing – upon gathering requirements, it helps by meeting expectations of users of that particular software
* Designing – helps in building a framework by outlining the software’s structure, navigation and database design
* Coding – this phase is where engineers and developers convert the software design into tangible code
* Testing – quality inspection and thorough examination of the software to eliminate bugs or glitches that might have slipped during coding
* Deployment – presentation the fine-tuned programme to its end-users
* Maintenance – constant assistance and improvement to improve on functionality of the software for its users

1. Compare and contrast the Waterfall and Agile methodologies. Provide examples of scenarios where each would be appropriate.

**Waterfall methodology,** as the name states, is called "waterfall" because the process flows sequentially, like a waterfall, through different stages of development. It is a traditional and linear approach to software development where each phase of the project must be completed before moving on to the next.

**Agile methodology,** is a flexible and iterative approach to software development that emphasizes collaboration, customer feedback, and small, rapid releases. Unlike the linear Waterfall methodology, Agile is designed to accommodate changes and evolving requirements throughout the development process.

When comparing the two methodologies;

|  |  |
| --- | --- |
| **Waterfall** | **Agile** |
| **Process structure** | |
| Linear and sequential. Each phase (e.g., requirements, design, implementation, testing) must be completed before moving on to the next. There is little overlap between phases. | Iterative and incremental. Work is divided into small cycles (sprints), with each cycle including planning, development, testing, and review. Different phases can overlap, allowing for flexibility and constant feedback. |
| **flexibility** | |
| Rigid. Changes are difficult and expensive to implement once a project is underway, especially in later stages. Requirements must be fully defined at the start. | Highly flexible. Changes to requirements can be made at any point during the project. Agile welcomes changing needs, even late in development. |
| **Customer involvement** | |
| Limited customer involvement after the initial requirements gathering. Customers usually see the final product after all development is complete. | Continuous customer involvement. Customers and stakeholders provide regular feedback after each sprint, ensuring that the product meets their expectations. |
| **Delivery** | |
| The product is delivered as a whole at the end of the development cycle. There is typically no usable product until the final phase. | The product is delivered incrementally, with small, functional parts released frequently. This allows for early and continuous delivery of value. |
| **Risk management** | |
| Higher risk. Problems might not be identified until late in the project, which can result in costly changes or missed deadlines. | Lower risk. Since work is done in iterations, risks are identified and addressed early, reducing the likelihood of major issues at the end of the project. |
| **Documentation** | |
| Emphasizes comprehensive documentation at each stage. Extensive documentation is created upfront, which guides the entire project. | Focuses on minimal necessary documentation. Agile values working software over detailed documentation, so only essential documents are maintained, with a focus on collaboration and adaptability. |
| **Team structure** | |
| Teams are typically more specialized, with separate groups responsible for different phases (e.g., design, development, testing). Each phase has clear handoffs. | Teams are cross-functional and collaborative, with all team members working together on various tasks (e.g., design, coding, testing) during each sprint. There is a greater emphasis on collaboration. |
| **Project Size and Scope** | |
| Best suited for projects with well-defined requirements and scope that are unlikely to change. It is often used in large-scale projects where thorough documentation is required (e.g., government projects, construction). | Ideal for projects with uncertain or evolving requirements. It works well in dynamic environments where quick delivery and responsiveness to change are crucial (e.g., startups, software development). |
| **Cost and Time Estimation** | |
| Easier to estimate cost and time upfront because the entire scope and requirements are defined at the beginning. However, changes can increase costs significantly. | More difficult to estimate cost and time upfront because requirements can evolve. However, Agile helps in managing scope creep by delivering incremental value and adjusting priorities regularly. |
| **Success Measurement** | |
| Success is measured by how closely the final product matches the initial plan, with less emphasis on flexibility and customer satisfaction during development. | Success is measured by the customer’s satisfaction and the delivery of functional software in small, usable increments, even if it diverges from the initial plan. |

- Waterfall is structured, predictable, and suitable for projects with clear requirements and minimal expected changes whereas Agile is flexible, adaptive, and best for projects where requirements are expected to evolve, and customer feedback is crucial throughout the development process.

- Choosing between the two methodologies depends on factors like project size, scope, customer needs, and the level of flexibility required.

1. Describe the roles and responsibilities of a Software Developer, a Quality Assurance Engineer, and a Project Manager in a software engineering team.

* In a software engineering team;

**A software developer** is responsible for designing, coding, testing, and maintaining software applications. They transform project requirements into functional code, using programming languages and development tools. Developers work closely with other team members to ensure the software meets the necessary specifications and quality standards.

**A Quality Assurance engineer** focuses on ensuring that the software product meets the desired quality standards. They design and execute tests to identify bugs, performance issues, and any deviations from requirements. QA engineers are involved throughout the development process, providing feedback to developers and ensuring that the final product is stable, reliable, and user-friendly.

**A project manager** oversees the planning, execution, and delivery of the software project. They coordinate tasks, manage timelines, allocate resources, and ensure communication across the team. The project manager also handles risk management, ensuring that the project stays on track and meets the client’s expectations. Their role is pivotal in ensuring that the project is completed on time and within budget.

1. Discuss the importance of Integrated Development Environments (IDEs) and Version Control Systems (VCS) in the software development process. Give examples of each.

* **An Integrated Development Environment (IDEs)** is a software application that provides a comprehensive environment for software development. It typically includes a code editor, a compiler or interpreter, a debugger, and build automation tools, all within a single user interface. **The importance of an IDE is** to streamline the development process by allowing developers to write, test, and debug code in one place. Popular IDEs include Visual Studio, Eclipse, and IntelliJ IDEA.
* **A Version Control System (VCS)** is a tool that helps track and manage changes to code over time. It allows multiple developers to work on a project simultaneously by keeping a history of all changes made to the codebase. **The importance of a VCS developer** is to revert to previous versions, compare changes, and manage conflicts that arise when merging code from different team members. Common version control systems include Git, Subversion (SVN), and Mercurial.

1. What are some common challenges faced by software engineers? Provide strategies to overcome these challenges.

* Keeping up with the rapidly changing technology, that evolves quickly, with new languages, frameworks, and tools emerging all the time. Staying up-to-date can be overwhelming.Continuous learning where engineers can allocate time to read tech blogs, attend workshops, complete online courses, actively participate in developer communities and mentorship can help.
* Dealing with technical debts, where quick fixes or shortcuts taken during development can lead to long-term maintenance issues, often called "technical debt." If one can prioritize refactoring and cleaning up code during sprints and implement best practices like code reviews, and automated testing it could prevent accumulating more technical debt.
* Managing large and complex codebases can be difficult to manage, leading to bugs, inefficiencies, and increased development time. Breaking the codebase into smaller, modular components and apply design patterns that promote reusability and maintainability can also help keep the codebase manageable.
* Debugging and Troubleshooting by finding bugs, especially in large, complex systems, can be time-consuming and frustrating. Having systematic debugging approaches, such as logging and using debuggers can also bring fresh perspectives that help identify and solve issues faster.
* Ensuring that software is secure from potential vulnerabilities can be complex and requires constant vigilance. Initiating the best security practices from the beginning, such as input validation, encryption, regular security audits and keeping up with the latest security threats ensures that the software is regularly patched

1. Explain the different types of testing (unit, integration, system, and acceptance) and their importance in software quality assurance

* Software Quality Assurance testing ensures that software meets the required standards and functions as expected. These types of testing can be manual or automated and are typically conducted in sequence, progressing from unit testing to acceptance testing. This includes:
* **Unit testing**

Purpose: Validates individual components (units) of the software.

Focus: Tests a single function or method in isolation.

* **Integration testing**

Purpose: Ensures that individual units or components work together as expected.

Focus: Tests the interaction between integrated units.

* **System testing**

Purpose: Verifies the complete and integrated software system to ensure it meets the specified requirements.

Focus: End-to-end testing of the application.

* **Acceptance testing**

Purpose: Validates that the software meets business requirements and is ready for deployment.

Focus: Ensures that the software works for the user in real-world scenarios.

**Introduction to AI and Prompt Engineering**

1. Define prompt engineering and discuss its importance in interacting with AI models.

Prompt engineering is the practice of asking, crafting questions or giving instructions for AI tools to get the best response or outcome.

Importance in interaction with AI models include;

* Maximization of Model Performance where a well-constructed prompt can direct the model to produce accurate, relevant, and coherent responses.
* Effective prompt engineering reduces the need for multiple iterations by clearly specifying what you want, minimizing time spent correcting misunderstandings.
* Encouraging creativity and exploration by crafting imaginative or open-ended prompts and users can encourage the AI to generate more creative, novel ideas.
* Controlling output style and tone with different prompts that can lead to different formats of output e.g. asking the model to "explain" something versus "summarize" can result in varied levels of detail and language use.
* Improving decision-making scenarios with carefully engineered prompts can lead the AI to consider various perspectives, list pros and cons, or offer structured advice.

1. Provide an example of a vague prompt and then improve it by making it clear, specific, and concise. Explain why the improved prompt is more effective.

* Example 1

**Vague Prompt:** Draw a picture of a cat

**Improved Prompt:** Draw a picture of a tabby cat sitting at the balcony with trees in the background, basking in the sun.

* Example 2

**Vague Prompt:** Explain hypertension

**Improved Prompt:** Explain the difference between Hypertensive Emergency and Hypertensive Urgency, including their causes, symptoms, and typical treatments. Provide examples of lifestyle changes that can help manage Hypertension.

Improved prompts are more effective by:

* Providing clarity and precisions
* Focused Problem Solving
* Efficiency by allowing quicker responses
* AI Integration with improved prompts