#Part 1: Introduction to Software Engineering

Software engineering and its importance in technology

Software engineering is a branch of engineering that applies engineering principles and methods to develop and maintain software. It involves a continuous process involving design, development, testing, deployment, and maintenance.

Software engineering enables the creation of software applications and systems responsible for various aspects of modern life, including communication, commerce, healthcare, etc.

Three key milestones in the evolution of software engineering

1. **Waterfall Methodology (1970**): This marked the beginning of modern software engineering. The Waterfall approach emphasized a linear and sequential development process, dividing it into phases like requirements gathering, design, implementation, testing, and maintenance.
2. **Agile Methodologies (2001)**: The Agile Manifesto emphasized flexibility, collaboration, and customer satisfaction, allowing for rapid adaptation to changing requirements. This approach made software development more responsive to user needs.
3. **DevOps and Continuous Integration/Continuous Deployment (CI/CD) (2008-2010):** The emergence of DevOps and CI/CD marked a significant shift towards automation, speed, and reliability in software engineering. By integrating development and operations teams and automating testing, building, and deployment, DevOps and CI/CD enabled faster time-to-market, improved quality, and increased efficiency.

Phases of the Software Development Life Cycle (SDLC)

1. **Planning**: Define project scope, goals, timelines, budget, and resources. Identify target audience and their needs.
2. **Requirements Gathering**: Collect and document detailed requirements through stakeholder interviews, surveys, and analysis.
3. **Analysis**: Break down complex requirements into smaller, manageable parts. Define system architecture and technical specifications.
4. **Design**: Create visual representations of the system, including user interfaces, data models, and system architecture.
5. **Implementation (Coding)**: Write the code according to the design specifications.
6. **Testing**: Verify the software meets requirements through various testing methods (unit, integration, system, acceptance).
7. **Deployment**: Release the software to production, making it available to end-users.
8. **Maintenance**: Provide ongoing support, fix defects, and make updates to ensure the software continues to meet user needs.

Waterfall vs Agile

**Comparison**

|  |  |
| --- | --- |
| Waterfall | Agile |
| Linear, sequential, and phase-gated | Iterative, incremental, and flexible |
| Requirements gathered upfront, with little room for change | Requirements evolve through continuous stakeholder feedback |
| Emphasis on planning, documentation, and predictability | Emphasis on collaboration, adaptation, and rapid delivery |
| Testing occurs after implementation | Testing integrated throughout development |

**Scenarios**

|  |  |
| --- | --- |
| Waterfall | Agile |
| Safety-critical systems (e.g., medical devices, aerospace) | Rapidly changing market requirements or user needs |
| Regulatory environments with strict compliance requirements | Complex, innovative, or exploratory projects |
| Small, well-defined projects with fixed requirements | Projects with high uncertainty or ambiguity |
| Legacy system maintenance with limited changes | Large-scale, distributed, or cross-functional teams |

**Examples**:

* Waterfall: Developing a medical device with strict regulatory requirements, where changes are costly and risky.
* Agile: Building a mobile app with rapidly evolving user expectations, requiring frequent updates and adaptations.

Roles and responsibilities of a Software Developer, QA Engineer, and PM

* A Software Developer is responsible for designing, developing, testing, and maintaining software applications. They write code, collaborate with cross-functional teams to identify and prioritize project requirements. They also participate in code reviews, contribute to the improvement of the codebase, and troubleshoot and debug software issues.
* A Quality Assurance Engineer focuses on ensuring software quality by developing and executing comprehensive testing plans. They identify, report, and track software defects and issues, collaborating with developers to reproduce and resolve defects.
* A Project Manager leads and coordinates software development projects from initiation to delivery. They define project scope, goals, timelines, budget, and resources, developing and managing project plans, schedules, and resource allocation. Project Managers facilitate communication among team members, stakeholders, and customers, identifying and mitigating project risks, issues, and dependencies.

IDEs and VCSs

IDEs are environments that allow developers to easily write and debug code. Some IDEs provide code suggestions, for example, VS Code.

VCSs are tools that keep track of the different versions of code that developers have written. They are typically used to return to a specific version of the code. A great example is Git.

Challenges and strategies for software engineers

The most common challenge I think software engineers face is bugs, bugs, bugs. The strategy I suggest to solving that is to have a rubber duck to talk to or just soliloquise.

Types of QA testing

* **Unit Testing**: Examines individual software components (units) in isolation to ensure they function correctly. Importance: Catches bugs early, reduces debugging time, and ensures individual components work as expected.
* **Integration Testing**: Verifies how different units work together to ensure seamless interactions. Importance: Exposes integration issues, ensures data flows correctly, and confirms components interact as expected.
* **System Testing**: Evaluates the entire software system to ensure it meets requirements and works as expected in a real-world environment. Importance: Identifies system-level issues, ensures software meets requirements, and verifies overall functionality.
* **Acceptance Testing**: Confirms the software meets user acceptance criteria, is usable, and meets business requirements. Importance: Ensures software meets user needs, verifies business requirements, and confirms software is ready for deployment.

These testing types are crucial in software quality assurance because they:

* Catch defects early, reducing debugging time and costs.
* Ensure individual components, integrations, and the overall system work as expected.
* Verify software meets requirements, user needs, and business goals.
* Provide confidence in software reliability, stability, and usability.
* Support continuous improvement and refinement of the software development process.

#Part 2: Introduction to AI and Prompt Engineering

Prompt engineering

Prompt engineering is all about crafting questions or instructions in a way that helps get the best responses from an AI model.

Examples of AI prompts

* **Vague prompt**: Tell me about cats.
* **Improved prompt**: Tell me about the origin of black cats.

The improved prompt is better because it is clear and specific and tells the AI exactly what it should produce as an answer.