Introduction To Software Engineering

Definition :

Software engineering is the process of creating, testing, and deploying computer programs to solve real-world problems while following a set of engineering principles and best practices. The area of software engineering takes a disciplined structured approach to software development, with the stated objective of enhancing quality, time, and budget efficiency, as well as ensuring systematic testing and engineering certification.

How does it differ from traditional programming :

Software engineering it is different from traditional programming in that it covers a larger range of activities, including requirements gathering, design, testing, and maintenance, with a strong emphasis on quality, dependability, and maintainability.

Software Development Life Cycle (SDLC):

The Software Development Life Cycle (SDLC) is a technique for developing software in a systematic and structured manner. It has multiple phases:

- **Planning**: Involves determining the project's scope, purpose, resources, and timeframe.
- **Requirements analysis:** Involves gathering comprehensive functional and non-functional needs from stakeholders.
- **Design:** Designing the system's architecture, including software and hardware specs.
- **Implementation (or Coding):** Creating code based on design documentation.
- **Testing:** Testing software to ensure it functions properly and without faults.
- **Deployment:** Involves releasing software to consumers.
- **Maintenance:** Post-deployment software updates and fixes bugs and new features.

Comparing Agile and Waterfall Models:

Waterfall Model:

- Using a linear and sequential strategy.
- Each phase must be completed before proceeding to the next.
- Optimal for projects with clear needs and few scope modifications.

- Example: construction projects.

Agile Model:

- Using an iterative and gradual approach.
- Values flexibility and consumer input.
- Continuous planning, execution, and evaluation.
- Optimal for projects with changing requirements.
- Example: Software startups.

Requirements Engineering:

It define, documents, and maintain software project requirements. It is critical because it ensures that the end product fulfills the demands and expectations of stakeholders. The procedure includes:

- Eliciting requirements from stakeholders.
- Specification: Detailed documentation of needs.
- Validation involves ensuring the criteria are valid and practical.
- Managing changes to requirements throughout time.

Software Design Principles:

Modularity is a crucial principle in software design, allowing for independent development, testing, and maintenance. This promotes maintainability by isolating changes to individual modules, as well as scalability by allowing the system to expand with the inclusion of additional modules.

Testing in Software Engineering:

Testing is essential for ensuring software quality and functioning. The levels of testing are:

- Unit Testing involves evaluating individual components or modules.
- Integration testing involves evaluating the relationship between integrated elements.
- System testing involves thoroughly evaluating the entire system.
- Acceptance testing involves validating software against user needs.

Version Control Systems:

VCS track and manage code changes across various versions. They are critical for communication and tracking changes in software development. Examples include:

- Git is a distributed VCS that permits branching and merging.

- Subversion (SVN) is a centralised version control system that is noted for its simplicity.
- Mercurial is a distributed version control system (VCS) that differs from Git in architectural decisions.

Software Project Management:

A software project manager oversees the planning, execution, and delivery of software projects. Responsibilities include:

- Planning and scheduling: Defining project milestones and timelines.
- Resource management: Allocating and managing resources.
- Risk management: Identifying and mitigating risks.
- Communication: Ensuring effective communication among stakeholders.

Software Maintenance:

Updating software after deployment to fix bugs, enhance performance, or provide new features. Types of maintenance include:

- Correction: Bug fixes.
- Adaptive: Changing software to fit a new environment.
- Perfective: Improving features and performance.
- Preventive: Update software to avoid future troubles.

Ethical Considerations in Software Engineering:

Ethical issues in software engineering include data privacy, intellectual property rights, and software misuse. Engineers can adhere to ethical standards by following professional codes of conduct, ensuring transparency, and prioritising user safety and privacy.

References:

1. Bridges, J. 2023. *Software Project Management: Why It's Different*. Available from: https://www.projectmanager.com/blog/software-project-management-why-its-different.
2. Clark, H. 2024. *The Software Development Life Cycle (SDLC): 7 Phases and 5 Models*. Available from: https://theproductmanager.com/topics/software-development-life-cycle/ [Accessed 9 June 2024].
3. Hamilton, T. 2024. *What is Software Testing?* Available from: https://www.guru99.com/software-testing-introduction-importance.html.
4. Hoogenraad, W. 2023. *What is the difference between software engineering and programming?* Available from: https://en.itpedia.nl/2019/07/12/wat-is-het-verschil-tussen-software-engineering-en-programmeren/.
5. Jędrzejko, K. 2023. *The Importance of Software Requirements Engineering in IT Projects*. Available from: https://softwaremind.com/blog/software-requirements-engineering-the-driving-force-behind-successful-and-efficient-it-projects/.
6. Lockhart, L. 2023. Agile vs. Waterfall: 10 Key Differences Between the Two Methods. *Float*. 15 February. Available from: https://www.float.com/resources/agile-vs-waterfall [Accessed 9 June 2024].
7. Maitray-Gadhavi. 2023. *Software Maintenance Guide for 2024: Importance, Types and Process*. Available from: https://radixweb.com/blog/why-software-maintenance-is-necessary.
8. *The 5 Essential Elements of Modular Software Design - GenUI*. n.d. Available from: https://www.genui.com/resources/5-essential-elements-of-modular-software-design.
9. Yasar, K. 2023. *software engineering*. Available from: https://www.techtarget.com/whatis/definition/software-engineering.
10. Lawton, G. 2020. *5 examples of ethical issues in software development*. Available from: https://www.techtarget.com/searchsoftwarequality/tip/5-examples-of-ethical-issues-in-software-development.
11. Sommerville, I. (2016). Software Engineering. Pearson.
12. Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.
13. Beck, K., et al. (2001). Manifesto for Agile Software Development. [Agile Manifesto](https://agilemanifesto.org/).