Teamwork 1 – Report

Team name: OpsForge

Team members: Mao Tamura (e2502119), Dominik Stiegler (e2502128), Dario Mathys (e2502127)

1. **What is software engineering? Explain what it includes?**

Engineering involves analyzing, designing, building, verifying, and managing technical or social systems. It requires answering key questions:

- What problem needs solving?
- What solution and characteristics will address it?
- How will the solution be designed and constructed?
- How will errors be detected and corrected?
- How will the system be maintained, adapted, and improved over time?

2. **How engineering process is divided into different phases? What activities are involved in each phase?**

Software engineering requires a defined development process with three main phases:

1. Definition Phase: Identify system requirements such as information, functions, performance, behavior, interfaces, constraints, and validation criteria. Key tasks: system/information engineering, project planning, and requirements analysis.
2. Development Phase: Specify data structures, architecture, implementation details, interfaces, and testing methods. Key tasks: software design, coding, and testing.
3. Maintenance Phase: Modify existing software for errors, environmental changes, new features, or reengineering. Types of maintenance:
    - Correction
    - Adaptation
    - Enhancement
    - Prevention

Complementary activities include reviews for quality, thorough documentation, and change control to manage and track modifications.

3. **Why do we need software development process models?**

We need software development process models because they provide structure, discipline, and guidance for building software in a predictable and efficient way. Specifically, they are needed to:

- Organize work: Define clear phases, tasks, and responsibilities so the team knows what to do and when.
- Manage complexity: Break down large, complex projects into manageable steps.
- Ensure quality: Incorporate reviews, testing, and validation to reduce errors.
- Control time and cost: Help estimate resources, schedules, and budgets.
- Support communication: Provide a common framework for developers, customers, and stakeholders to understand progress and expectations.
- Handle change: Allow systematic adaptation when requirements or environments evolve.

4. **How Spiral model differs from Waterfall model?**

- Waterfall is linear and sequential, Spiral is iterative and cyclic.
- Waterfall is rigid (hard to go back), Spiral is flexible (phases can be revisited).
- Waterfall has little focus on risk, Spiral explicitly emphasizes risk analysis.
- Waterfall assumes stable, well-defined requirements, Spiral allows evolving requirements.
- Waterfall fits small/low-risk projects, Spiral suits large/complex/high-risk projects.

5. **What are the advantages of Modified Waterfall model over Pure Waterfall model?**

- In Modified Waterfall, you can go back to earlier phases if needed; Pure Waterfall does not allow this.
- Mistakes discovered later can be fixed by revisiting previous stages, reducing risk of failure.
- It handles changes in requirements better than the rigid Pure Waterfall.
- Continuous verification between phases ensures fewer defects.

6. **What are the advantages of iterative development?**

- A working version of the software is available early, giving value to users sooner.
- Requirements can be refined and adapted as the project progresses.
- Problems are identified earlier because testing and feedback occur in each iteration.
- Users and stakeholders can review prototypes regularly, ensuring the product meets expectations.
- Continuous testing and refinement lead to fewer defects.
- Breaking the project into smaller iterations makes planning, tracking, and controlling progress simpler.

7. **What are the advantages of RUP over other development process models?**

- Like Spiral, RUP develops the system in cycles, reducing risks and allowing gradual refinement.
- Risks are identified and mitigated early in each iteration.
- Requirements are captured through use cases, ensuring the system aligns closely with user needs.
- Strong focus on building a robust, well-defined architecture early, which improves scalability and maintainability.
- Can adapt to changing requirements more easily than rigid models like Waterfall.
- Continuous testing and reviews throughout the lifecycle improve software reliability.
- Defines responsibilities for team members, improving coordination and project management.