**Assignment – 02[04/06/2024]**

**Task 1:SDLC Overview - Create a one-page info graphic that outlines the SDLC phases (Requirements, Design, Implementation, Testing, Deployment), highlighting the importance of each phase and how they interconnect.**

**SDLC(Software Development Life Cycle):**

The software development life cycle (SDLC) is the process of planning, writing, modifying, and maintaining software. SDLC aims to minimize project risks through planning so that software meets customer expectations during production and beyond.Software development can be challenging to manage due to evolving requirements, technology upgrades, and multi-disciplinary collaboration. The software development life cycle (SDLC) methodology offers a structured management framework with specific deliverable at each phases of the software development process.

**Benefits of SDLC:**

* Increased visibility of the development process for all stakeholders involved
* Efficient estimation, planning, and scheduling
* Improved risk management and cost estimation
* Systematic software delivery and better customer satisfaction.

The software development life cycle (SDLC) outlines the tasks required to build a software application. The development process involves several stages as developers add new features and fix bugs in the software.

Requirement

Design

Implementation

Testing

Deployment

**1.Requirements Phase:**

**Importance:**

The requirements phase involves decisively gathering and documenting the software requirements from stakeholders. This phase unequivocally sets the foundation for the entire project by explicitly defining what needs to be built and the problem it solves.

**Interconnection:** The requirements gathered in this phase indisputably serve as input for the design phase. Unambiguously clear and comprehensive requirements ensure that the subsequent phases stay aligned with the intended goals of the project.

**2.Design Phase:**

**Importance:**

In the design phase, the system architecture, software architecture, and detailed design are adamantly and confidently created based on the gathered requirements. This phase resolutely focuses on how the system will be structured and how various components will interact with each other.

**Interconnection:** The designs developed in this phase are assertively used as a blueprint for implementation. A well-designed system ensures that the software is indubitably scalable, maintainable, and meets the functional and non-functional requirements defined in the previous phase.

**3.Implementation Phase:**

**Importance:**

The implementation phase decisively involves actual coding or programming based on the designs created in the previous phase. Developers indomitably write, compile, and integrate code to build the software system.

**Interconnection:** The implementation phase forcefully relies heavily on the designs provided in the previous phase. Developers confidently refer to design documents and translate them into executable code. Continuous and unwavering communication with the design team ensures that the implementation assertively stays aligned with the intended architecture and functionalities.

**4.Testing Phase:**

**Importance:**

Testing is crucial for ensuring that the software undeniably meets quality standards and performs as expected. It uncompromisingly involves identifying defects, errors, or bugs in the software and fixing them before deployment.

**Interconnection:** The testing phase relentlessly validates the software against the requirements specified in the first phase. Test cases are decidedly derived from the requirements and design documents. Feedback from testing assertively informs developers about necessary adjustments or fixes, which may mandatorily require updates to requirements or design.

**5.Deployment Phase:**

**Importance:**

The deployment phase evidently involves releasing the software for production use. It steadfastly includes activities such as installation, configuration, and deployment of the software to end-users.

**Interconnection:** The deployment phase unequivocally relies on the successful completion of previous phases. The deployed software should unmistakably meet the requirements defined in the first phase, have a robust design, be assertively implemented, and be thoroughly tested to ensure reliability and usability.

**Task 2: Research and compare SDLC models suitable for engineering projects. Present findings on Waterfall, Agile, Spiral, and V-Model approaches, emphasizing their advantages, disadvantages, and applicability in different engineering contexts.**

Overview of different software development life cycle (SDLC) models:

1. **Waterfall Model:**

**Advantages:**

* Simple and easy to understand.
* Well-suited for projects with clear and stable requirements.
* Each phase has specific deliverables, making it easy to measure progress.

**Disadvantages:**

* Lack of flexibility, difficult to accommodate changes once the development process has started.
* High risk of late-stage changes or errors due to limited customer involvement until the end.

Applicability: Suitable for projects where requirements are well-defined and unlikely to change significantly, such as in traditional engineering projects like construction or manufacturing.

**2.Agile Model:**

**Advantages:**

* Highly flexible and adaptable to changing requirements.
* Customer involvement throughout the development process ensures alignment with user needs.
* Iterative approach allows for early and continuous delivery of working software.

**Disadvantages:**

* Requires active involvement and collaboration from all stakeholders, which may not always be feasible.
* May lack documentation, making it challenging for new team members to understand the project.

Applicability: Ideal for engineering projects with evolving requirements, where rapid development and frequent feedback are critical, such as software development or product prototyping.

**3.Spiral Model:**

**Advantages:**

* Risk management is integrated into the development process, allowing for early identification and mitigation of risks.
* Highly adaptable, allowing for iterative development and refinement.
* Suitable for large and complex projects where requirements are uncertain or evolving.

**Disadvantages:**

* Can be time-consuming and costly due to its iterative nature.
* Requires extensive documentation and planning, which may not be feasible for smaller projects.

Applicability: Well-suited for engineering projects with high technical or operational risks, such as aerospace or defense projects, where risk management and iterative development are crucial.

**4.V-Model:**

**Advantages:**

* Emphasizes verification and validation activities throughout the development lifecycle, ensuring higher-quality deliverables.
* Provides a structured approach with clear deliverables for each stage of development.
* Well-suited for projects with strict regulatory or compliance requirements.

**Disadvantages:**

* It can be rigid and sequential, making it challenging to accommodate changes or feedback. This may result in longer development cycles due to the sequential nature of activities.

Applicability: Commonly used in engineering projects with stringent quality assurance requirements, such as medical device or automotive engineering, where thorough testing and validation are essential.

* Each SDLC model has its strengths and weaknesses, and the choice depends on factors such as project requirements, complexity, and timeline.