**Frimpong Prince**

**BSc. Computer Engineering**

**COE 454: Software Engineering**

**Waterfall Model**

This SDLC model is the oldest and most straightforward. With this methodology, we finish one phase and then start the next. Each phase has its own mini-plan and each phase “waterfalls” into the next. The phases include Requirements, Design, Implementation, Testing, Deployment, and Maintenance.

* Problem Definition: The problem is clearly defined at the beginning of the project.
* Requirements Gathering and Analysis: Detailed requirements are gathered and analyzed.
* Planning and Designing: A comprehensive plan and system design are created based on the requirements.
* Development/Implementation: The system is developed according to the design specifications.
* Testing: The developed system undergoes rigorous testing to ensure it meets the specified requirements.
* Deployment: The tested system is deployed to the production environment.
* Maintenance and Support: The deployed system is maintained and supported to ensure its continued operation and relevance.

**Agile Model**

Agile focuses on iterative development, where requirements and solutions evolve through collaboration.

* Problem Definition: The problem is defined in a general sense, allowing for flexibility.
* Requirements Gathering and Analysis: Requirements are gathered iteratively, allowing for continuous feedback and refinement.
* Planning and Designing: Planning and design happen in small, iterative cycles (sprints), allowing for adaptability.
* Development/Implementation: Development occurs in short cycles, with each iteration delivering a potentially shippable product increment.
* Testing: Testing is integrated into each iteration, ensuring continuous verification of the product.
* Deployment: Deployment can occur at the end of each iteration, providing incremental delivery.
* Maintenance and Support: Continuous maintenance and support are provided, often integrated within the iterative cycles.

**V-Model (Verification and Validation Model)**

An extension of the waterfall model, this SDLC methodology tests at each stage of development. It emphasizes verification and validation processes and maps each development phase directly to a testing phase.

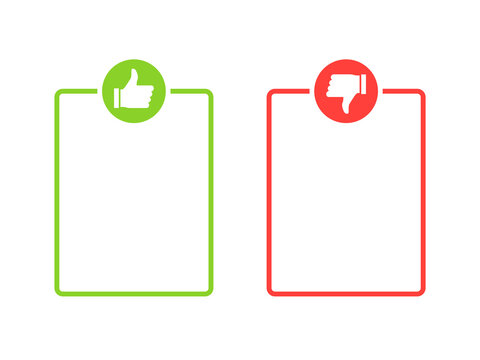
* Problem Definition: The problem is clearly defined and understood at the outset.
* Requirements Gathering and Analysis: Requirements are gathered and analyzed with an emphasis on validation.
* Planning and Designing: Planning and design phases are meticulously executed, with a focus on creating test plans alongside the design.
* Development/Implementation: The system is developed following the detailed design specifications.
* Testing: Testing phases are mapped to each development phase, ensuring each component is validated against its requirements.
* Deployment: The validated system is deployed to the production environment.
* Maintenance and Support: The deployed system undergoes continuous maintenance and support, with a focus on validation and verification.

**Spiral Model**

One of the most flexible of the SDLC models is the spiral model. It resembles the iterative model in its emphasis on repetition. Even this model goes through the planning, design, build and test phases again and again, with gradual improvements at each stage

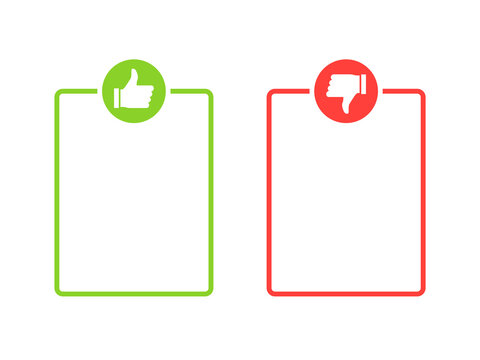
* Problem Definition: The problem is initially defined, with a focus on identifying risks.
* Requirements Gathering and Analysis: Requirements are gathered and analyzed iteratively, with continuous risk assessment.
* Planning and Designing: Planning and design occur in iterative cycles, each addressing identified risks.
* Development/Implementation: Development is done in small increments, with each cycle building on the previous one.
* Testing: Testing is conducted in each iteration, ensuring that risk mitigation strategies are effective.
* Deployment: The system is deployed incrementally, following thorough testing in each cycle.
* Maintenance and Support: Ongoing maintenance and support are provided, with continuous risk assessment and management.

**Waterfall Model**



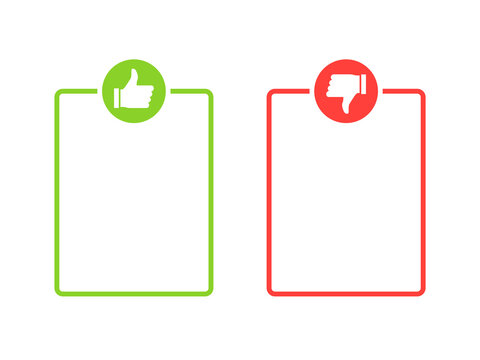
* Inflexible to changes.
* Late discovery of issues.
* Not suitable for complex or long-term projects.
* Simple and easy to understand.
* Clear milestones and deadlines.
* Well-documented process.

**Agile Model**



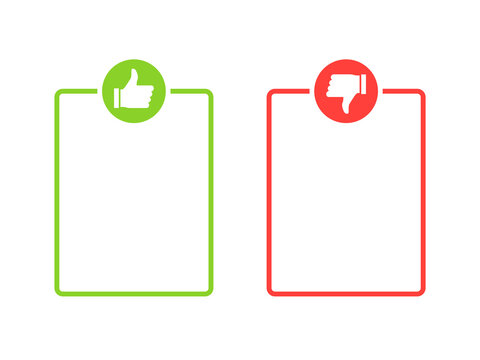
* Requires experienced team.
* Can be resource-intensive.
* Less predictable outcomes.
* Flexibility and adaptability.
* Continuous customer feedback.
* Faster delivery of functional software.

**V-Model (Verification and Validation Model)**



* Inflexible to changes.
* High dependency on initial requirements.
* Can be costly and time-consuming.
* Emphasis on testing and validation.
* Clear and structured approach.
* Easier defect tracking.

**Spiral Model**



* Complex to manage.
* Can be expensive.
* Requires expertise in risk assessment.
* Focus on risk management.
* Iterative refinement.
* Suitable for large and complex projects.

When each methodology is suitable

* Waterfall Model: Best for projects with well-understood requirements, minimal changes, and a need for thorough documentation.

Examples: Government projects, manufacturing projects, and projects with strict regulatory requirements.

* Agile Model: Ideal for projects requiring flexibility and customer involvement.

Examples: Web applications, and any project where requirements evolve over time.

* V-Shaped Model: Suitable for projects with clear and fixed requirements, where validation and verification are crucial.

Examples: Medical device software, aerospace projects, and projects requiring high levels of testing and compliance.

* Spiral Model: Best for large, complex projects with significant risks and evolving requirements, requiring continuous risk assessment and flexibility.

Examples: Large-scale enterprise applications, telecommunications projects, and any project with evolving requirements and high risk.