**Assignment 1**

**SDLC & Agile**

Alishba Azhar: 000925324   
Thomas Coons: 000317961   
Pouya Tayyari: 000949516   
Khushal Parimal Valand: 000922744

Mohawk College

COMP-CO835-09: Object Oriented Systems

Dr. Stanley Ukah

September 15th, 2024

1. **Introduction to Software Development Methodologies**Methodologies for software development are organized frameworks that are used to organize, manage, and carry out the process of creating software applications. Throughout the software development lifecycle (SDLC), from early planning to final use and maintenance, these approaches give programmers standards to follow.   
     
   Some of the methodologies are:   
   * **Waterfall** - a precise, ordered procedure in which each stage must be finished before going on to the next.
   * **Agile** - a flexible, ongoing procedure that prioritizes regular releases and user input.
   * **Iterative** - a process in which ideas are divided into segments and then repeated and improved continuously until the result satisfies specifications.

Selecting an appropriate methodology is essential for system analysis and design since it influences quality, communication, costs, and timeliness.

1. **Traditional Systems Development Life Cycle**The phases of the Traditional Systems Development Life Cycle (SDLC) include Analysis, Design, Implementation, and Maintenance.   
   * Planning: Determining the issue and getting consent.
   * Analysis: Recognizing and recording requirements and business demands.
   * Design: Assembling system parts to fulfil the needs that have been determined.
   * Implementation: System development, testing, and integration.
   * Maintenance: Setting up and looking after the system.

**Characteristics of Traditional SDLC:**

* Fixed Standards: It is challenging to apply modifications to requirements once they have been defined.
* Emphasis on documentation: Every step of the way, especially in the analysis and design stages, calls for extensive documentation.

**Advantages of Traditional SDLC**:

Having distinct phases and standards makes it easier to manage. It works effectively for tasks with clear specifications.

**Disadvantages of Traditional SDLC**:

It is less adaptable because it is difficult to implement modifications once a phase is finished.

**Projects Suitable for SDLC:**

* Healthcare Systems: Because of their crucial nature, systems like electronic health records (EHR) require extensive documentation and a methodical, tried-and-true methodology.
* Banking Systems: These include apps for banking and insurance that handle sensitive data and must strictly conform to rules.

1. **Object-Oriented SDLC**

**The principles of Object-Oriented SDLC are:**

* + Conception
  + Initiation
  + Analysis
  + Design
  + Construction
  + Testing
  + Deployment

It is important to understand these steps are not sequential and can run concurrently with other phases. This is the core feature of Object-Oriented SDLC. The use of agile methodologies makes it more flexible and dynamic (Ukah, n.d., slides 19–21)

**Compare and contrast Object-Oriented SDLC with Traditional SDLC:**

The main difference between the two System Development Life Cycles is how they are approached. Object Oriented SDLC requires agile and less rigorous methods of development. It allows for change and can quickly adapt (Satzinger et al., 2016, p. 8). Tradition SDLC is developed via the Water Fall methodology. It assumes a step-by-step process where one process cannot start before the previous process has finished (Satzinger et al., 2016, p. 299).

**Scenarios where Object-Oriented SDLC is Preferred:**

Object-Oriented SDLC is preferred when we are developing projects that are in line with   
the principles of Object-Oriented SDLC. When the main goal is to have working software that is high-quality, we should use OO SDLC. We also use OO SDLC when we are expected to embrace change and to allow for quick changes throughout the development process (Satzinger et al., 2016, p. 306).

1. **Agile Development**

Agile development is a project management technique used in collaborative software development environments to deliver small, functional and flexible changes throughout the System Development lifecycle.

Iterative Development is an approach that develops the system in small divisions (usually between 1-4 weeks) that are usable products. In this process, the system receives feedback in each iteration, the system receives feedback which eventually results in a complete product.

Customer Collaboration in the agile process the emphasis is on getting constant feedback from stakeholders. Customers, or the end users are the most important stakeholder in this process. Therefore, having customer collaboration is a critical part of agile development methodology.

Responding to Change as it was mentioned before, Agile is a process that needs to be flexible to changes. Therefore, being able to adopt responsive changes is a critical part of this methodology, even in later stages of the process.

**Popular Agile Methodologies:**

**Scrum**:   
  
In this methodology's tasks are divided into few iterations that are usually due between 1-4 weeks. With Scrum you are obligated to deliver within the specified time. Big tasks are divided into smaller tasks to make it manageable for the team. A daily scrum meeting is when different teams meet up to discuss their process and the obstacles they are facing. Scrum has three defined roles. Product Owner, Scrum Master and development team are the main roles in this process.  

**Kanban**:

This methodology is based on visualizing workflow structure that keeps the team moving along the agile process. In Kanban updates are released once they are ready, the Kanban does not follow a particular timeline. But there are important metrics that evaluate how Kanban works. For example, The Cumulative Flow Diagram helps identify different bottlenecks in the process. Kaban Workflow can change at any time, it can be adaptive to work process limit and change according to the number of team members.  

In the end, the agile method is compatible with rapid changes, which can be very beneficial in an industry that is changing rapidly. It also provides a faster-to-market strategy that pushes the product forward into the market to capture market share faster. On the other hand, it can sometimes leave team members with Unclear Requirements. Lack of product vision can lead to scope creep and extended deadlines.

1. **Comparative Analysis**

When it comes to a comparative analysis between different SLDC methodologies, various factors affect choosing the right one for your project. These factors include flexibility, adaptivity, risk management, stakeholder involvement and time to market.

I. **Traditional SLDC (Waterfall)**

Flexibility: The traditional SDLC follows a rigid, sequential process. Once a phase is completed, revisiting it to accommodate changes is difficult. Flexibility is limited because project requirements are fixed at the start.

Adaptability: The Waterfall model does not handle evolving project requirements well. It assumes that all needs are understood upfront, making it unsuitable for projects where requirements change frequently.

Risk Management: Risks are addressed in the early stages. However, since testing happens late in the process, there’s a higher risk of discovering critical issues later in development.

Stakeholder Involvement: Stakeholders are most involved at the start and end phases (requirements gathering and final delivery), but their involvement is minimal during development.

Time-to-Market: Due to its linear structure, the Waterfall model often has a longer time-to-market. Each phase must be completed before moving on to the next, delaying delivery.

Project Suitable for SDLC:

* Healthcare System: Requires extensive documentation and precise requirements
* Banking Systems: Strict Regulatory adherence makes the rigid structure more suitable.

II. **Object-oriented SDLC**

Flexibility: Object-Oriented SDLC allows greater flexibility as it incorporates iterative cycles. Changes can be integrated throughout the development process.

Adaptability: Highly adaptable due to its iterative approach. Developers can revisit earlier phases as new insights arise, making it suitable for projects with changing requirements.

Risk Management: Risks are addressed throughout multiple iterations, allowing for frequent adjustments. Early testing within each cycle helps catch issues earlier in the process.

Stake Involvement: There’s more opportunity for ongoing stakeholder involvement compared to Traditional SDLC. Regular feedback is integrated as new iterations are developed.

Time-to-Marker: Faster than Traditional SDLC, as some components can be developed parallel and iteratively, reducing the overall time to release.

Projects Suitable for Object-Oriented SDLC:

* Complex Systems: Where modularity and flexibility are key.
* Quality-Driven Projects: focuses on maintaining high standards and embracing change.

III. **Agil Development**

Flexibility: Agile is the most flexible methodology. It allows continuous adaptation to changes, even late in development. Requirements are expected to evolve.

Adaptability Agile thrives in environments of rapid change. It embraces iterative development, which allows teams to adjust to new inputs continuously.

Risk Management: Risks are mitigated through constant iteration and feedback. Each sprint provides an opportunity to address issues as they arise, reducing the chance of major problems going unnoticed.

Stakeholder Involvement: Agile places heavy emphasis on stakeholder involvement. Clients and users provide feedback during each iteration, ensuring the project aligns with expectations.

Time-to-Market: Agile enables rapid delivery by breaking projects into small, manageable sprints. Products can often launch earlier with incremental improvements.

**Projects Suitable for Agile**:

* Startups and Innovative Projects: Where market changes and customer needs evolve quickly.
* Customer-Focused Applications: Continuous feedback and iteration help meet changing user demands.

**Summary of Key Findings**

* Flexibility and Adaptability: Agile excels in both areas, making it the best choice for projects requiring ongoing adaptation. Object-Oriented SDLC also performs well, while Traditional SDLC is the least flexible.
* Risk Management: Agile offers the most robust risk management through continuous feedback, while Object-Oriented SDLC provides moderate risk control. Traditional SDLC addresses risks early but may miss problems until later stages.
* Stakeholder Involvement: Agile is highly collaborative, with frequent stakeholder input, whereas Traditional SDLC limits involvement after the initial planning phase.
* Time-to-Market: Agile is designed to accelerate delivery, followed by object-oriented SDLC. Traditional SDLC has the slowest time-to-market due to its linear nature.

**Recommendations**

* Agile Development is the best suited for projects that require fast adaptation, frequent stakeholder input, and rapid delivery, such as startups and consumer-facing applications.
* Object-Oriented SDLC is ideal for projects where high-quality software is crucial, some flexible is needed, such as modular, large-scale systems.
* Traditional SDLC works best for projects with fixed, well-defined requirements that require extensive documentation, such as regulatory systems in healthcare or banking

**References**

Atlassian. (n.d.). *What is agile?* <https://www.atlassian.com/agile>

Satzinger, J. W., Jackson, R. B., & Burd, S. D. (2016). *Systems analysis and design in a changing world*

(7th ed.). Cengage Learning.

Ukah, S. (n.d.). *Object-oriented systems* [PowerPoint slides]. Mohawk College.

<https://mycanvas.mohawkcollege.ca/courses/106931/files/20527405?module_item_id=5749263>