**Day 2 Assignment 3: 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.**

**SDLC:**

Software development existence cycle models are frameworks that manual the development of software program tasks from start to completion. There are several software development existence cycle fashions, every with its personal set of benefits and drawbacks. In this answer, we will compare some of the maximum popular software development life cycle fashions, inclusive of the Waterfall version, the Agile version, and the Spiral version.

You can see the most used and reliable SDLC models on the list below:

* [Waterfall model](https://en.wikipedia.org/wiki/Waterfall_model)
* [Iterative model](https://en.wikipedia.org/wiki/Iterative_and_incremental_development)
* [Spiral model](https://en.wikipedia.org/wiki/Spiral_model)
* [V-shaped model](https://en.wikipedia.org/wiki/V-Model_(software_development))
* [Agile model](https://en.wikipedia.org/wiki/Agile_software_development)

**Different SDLC Methodologies:**

**Waterfall Model:**

The Waterfall model, a classic and well-established SDLC methodology, offers a structured and linear approach to project development. It progresses through distinct phases, each building upon the completion of the previous one. These phases include requirements gathering, system design, implementation, testing,deployment, and maintenance (Royce, 1970). The Waterfall approach is particularly suitable for projects with well-defined and stable requirements, where changes are expected to be minimal. However, its rigid

structure makes accommodating changes difficult once a phase is completed.

**Iterative Sdlc Model:**

The iterative model resembles a waterfall model, but there is quite a considerable difference between them. For example, let’s suppose there’s an app that contains ten core features. In the waterfall case, all ten functions will be thoroughly planned during the requirement analysis and design phases and then steadily implemented during the development stage. The iterative model is quite different. It implies that the whole process is divided into a particular number of iterations, and during each of them, developers build a limited number of features.

So, the Iterative SDLC model does not require a complete list of requirements before the project starts. The development process may start with the requirements to the functional part, which can be expanded later. The process is repetitive, allowing to make new versions of the product for every cycle. Every iteration (that lasts from two to six weeks) includes the development of a separate component of the system. After that, this component is added to the features developed earlier. Speaking with math terminology, the iterative model is a realization of the sequential approximation method; that means a gradual closeness to the planned final product shape.

Use cases for the Iteration model:

* The requirements for the final product are clear from the beginning
* The project is large and includes complex tasks
* The main task is predefined, but the details may change in the process

**Spiral SDLC Model:**

Spiral model is a combination of the Iterative and Waterfall SDLC models with a significant accent on the risk analysis. The main issue of the spiral model is defining the right moment to take a step into the next stage. The preliminary set timeframes are recommended as the solution to this issue. The shift to the next stage is done according to the plan, even if the work on the previous step isn’t done yet. The plan is introduced based on the statistical data received in the last projects and even from the personal developer’s experience.

Use cases for the Spiral model

* The customer isn’t sure about the requirements
* Significant edits are expected during the software development life cycle
* Risk management is highly essential for the project

This approach results in constant learning, meaning that during each iteration, the team makes observations and brings new ideas to the next iteration.

**V-shaped SDLC Model:**

The V-shaped algorithm differs from the previous ones by the work approach and the architecture. If we visualize this model, we’ll see that there appears one more axis, unlike the waterfall and iterative models. Along with the first one, they constitute the V letter.

The V-model is called this way because of the scheme’s appearance and because its primary priorities are Verification and Validation. Stages positioned along the left axis display the verification phases, and the ones on the right are responsible for validation.

Use cases for the V-shaped model:

* For the projects where accurate product testing is required
* For the small and mid-sized projects, where requirements are strictly predefined
* The engineers of the required qualification, especially testers, are within easy reach

**Agile SDLC Model:**

Agile is a philosophy, not a specific development approach. It is a whole family of methodologies. Scrum, Kanban, or XP (extreme programming) are among the most common realisations of the Agile SDLC. Let’s find out the core principles of Agile in general and then take a brief look at some of its realisations.

Its first peculiarity is that all work is split into iterations like the iterative model. These iterations are named sprints. The team initially defines what actions they’ll need to perform in a particular timeframe. The main difference with the iterative approach is that this amount of work is not strict and can be changed in the middle of the process.

Use cases for the Agile model:

* The users’ needs change dynamically
* Less price for the changes implemented because of the numerous iterations
* It requires only initial planning to start the project

**Scrum:**

Sprints lie at the core of Scrum. A sprint is an iteration that lasts from 2 to 4 weeks. It starts with the planning of the product and an iteration itself. Then, the product passes the stages of a chosen software development methodology (XP is the most popular option in terms of Scrum). At the end of the sprint, the team delivers the developed part of the product, and then it’s time for the retrospective – sprint review and analysis.

The core methods of Scrum contain sprint planning, daily meet-ups, demos and reviews, and a retrospective at the end of each sprint. Its main principles are сourage, commitment, focus, openness, respect.

Also, Scrum, as a realization of Agile, is subject to 12 Agle principles.

**Kanban:**

Kanban is another realisation of Agile. It somehow reminds Scrum, but still, they have some distinct differences. Both approaches use boards to track progress and have similar sections. The ones of Kanban are:

* Requested
* In progress
* Done

**CONCLUSION:**

Methodologies eliminate the need to invent new management and development techniques. They are a framework for the software development process. Also, they provide every team member with a clearly defined plan so that everyone understands what they’re doing, why, and what’s the final goal.

Different models were created during the SDLC evolution to meet a wide variety of development requirements and expectations. Nowadays, the world is moving towards flexibility instead of excessive regularity and orderliness. Still, agile approaches perfectly combine responsiveness and well-organized project management.