**Key Features/Principles**

* **Agile** – adaptation on UI and UE feedback with the overall purpose of customer satisfaction and deliverance
* **Kanban** - focuses on visualizing tasks, limiting work in progress, and ensuring a smooth flow of tasks from start to finish.
* **Scrum** - a set of meetings, tools, and roles that work in concert to help teams structure and manage their work. Six key principles that sculpt the workflows of your team: Empirical Process Control, Self-Organization, Time-Boxing, Value-Based Prioritization, Iterative Development, and Collaboration.
* **Waterfall** – unbreakable workflow order through analysis, design, implementation, testing and maintenance.

**Similarities/Differences**

The Waterfall is suitable method for well-defined projects, whilst Agile, including Scrum and Kanban, is more adaptive and centred towards customers. Scrum is ideal for projects with fixed-length iterations, while Kanban is better for continuous delivery and flow-based work. Kanban shares a key principle with Scrum and Agile - continuous development, delivering change in small manageable increments.

Agile and Waterfall have similarities as both approaches aim to deliver a final product without defects and meet quality standards. Agile and Waterfall employ project management tools to track progress. Tools like Gantt charts, Kanban boards, and various software applications can be adapted to fit either methodology. Although they do have their differences, Agile promotes ongoing collaboration, while Waterfall focuses on planning and execution in predetermined stages.

|  |  |  |
| --- | --- | --- |
| **METHOD** | **STRENGTHS** | **WEAKNESSES** |
| **Agile** | • Constant improvement and quality assurance  • More flexible to change needs  • Hands-on approach for customers  • Faster delivery times on initial value  • Increased internal interactions | • Lack of long-term planning  • No clear picture of costs upfront (difficult planning)  • Lack of cross professionally trained teams available  • Problems with workflow coordination |
| **Kanban** | • Increased internal interactions, easy to use, can work in many different industries, can be customized, Encourages collaboration  • Task clarity - Equip team members with everything they need  • Status alignment, Improved team efficiency, better focus & avoid burnout | • Can be less useful in a dynamic setting  • Does Not show the timeframes  • Complexity limitations, Updates are essential, Project schedule management |
| **Scrum** | • Help teams complete project deliverables quickly & efficiently  • Ensures effective use of time and money  • Large projects are divided into easily manageable sprints  • Developments are coded and tested during the sprint review  • Works well for fast-moving development projects  • Team gets clear visibility through scrum meetings  • Adopts feedback from customers and stakeholders  • Short sprints enable changes based on feedback more easily | • Often leads to scope creep, due to the lack of a definite end-date  • The chances of project failure are high if individuals aren't very committed or cooperative  • Adopting the Scrum framework in large teams is challenging  • Framework can be successful only with experienced team members  • Daily meetings sometimes frustrate team members  • If any team member leaves in the middle of a project, it can have a huge negative impact on the project  • Quality is hard to implement until the team goes through an aggressive testing process |
| **Waterfall** | • Provides a way for large or changing teams to work together toward a common goal defined in the requirements phase  • Ensures a disciplined & structured organisation  • Provides a simple method to understand, follow & arrange tasks  • Facilitates management control and departmentalization based on deadlines  • Establishes good coding habits by defining first, then implementing design  • Provides easy access to early system design and specification changes  • Defines milestones and deadlines clearly | • Design flaws, when discovered, often mean starting over from scratch  It doesn’t incorporate mid-process feedback from users or clients and makes changes based on results.  • Delaying the testing until the end of development is common  • There’s no consideration for error correction  • The model doesn’t accommodate changes, scope adjustments, and updates well  • Work on different phases doesn’t overlap, which reduces the efficiency  • Projects don’t produce a working product until later stages  • Not an ideal model to use for complex and high-risk projects |

**Give an example of when each methodology is most suited. Include information on:**

**• Project and product types**

**• Requirements and documentation**

**• Compliance and governance**

**• Visibility and scope**

**• Plan and timescales**

**• Staffing and resources**

**• Volatility**

In the instance of having to develop a mobile app for a client, the most suited methodology to use would be Agile. I have chosen this methodology because it can provide the client with constant improvement and quality assurance as well as being more flexible. Mobile apps have specific due dates upon release, businesses need to know that the application has met its requirements to function as contracted. In addition, test applications can be used to acquire user feedback from User-Interaction hence why Agile can responsively help the workflow accumulate real time data upon testing our product, which allows us to notify our client with the project’s performance.

For a complex airline reservation system, I would use Kanban methodology as it is better for continuous delivery and flow-based work. Kanban focuses on visualizing tasks, limiting work in progress, and ensuring a smooth flow of tasks from start to finish allowing any requests to be dealt with fast and efficiently.

To develop a new video game, I am going to use Scrum as it is ideal for projects with fixed-length iterations and continuous development, delivering change in small manageable increments as it adopts feedback from customers and stakeholders. Communication becomes through [Scrum meetings](https://www.simplilearn.com/how-to-conduct-on-target-daily-scrum-meeting-article) and large projects are divided into easily manageable sprints ensuring effective use of time and money.

A customer support ticket system would pair well with the Waterfall method. Requirements, design, implementation, verification, and maintenance become tools to deliver the final product. Mainly Events would use a ticketing system and since some events can spring up, immediate ticket distribution is needed hence a customer support ticket system would become a necessary function to deal with customer/user enquires and questions. All stages proposed by the Waterfall method can be executed through accurate planning and a structured predetermined strategy.

**-Identify the process of requirements capture. You should include:**

Business Requirements - business what the stick or steak holders want you to do i.e., solving problems, building applications, algorithm, data handling, new features.

Functional Requirements - features based on businesses and technology combined.

software - or a specification. the blueprint of the project, this is for the developers to use for meeting their accomplishments.

Non-Functional Requirements –

* Roles and responsibilities during requirements capture.
* Backlog processes such as grooming and refinement.

**-Describe the following:**

* Epics – is a collection of required stories, they are useful when deciding what team member can work on what task. a piece of functionality that takes a longer amount of time to complete/ 3 months/project is split into task. an example is micro transaction split down so one task would be screen interaction, the other can be data handling.

- in agile a sprint is a task that is completed in 3-4 weeks, with planning and individual separate tasking, this is then tested! and feedback is assimilated

- enabler tasks are carried out through sprint to refactor your code/ organise it. i.e., making a piece of software more compliant (highest level) can take up to 6 moths for completion (group of related tasks)

* Features - a sprint of 1 week. 3 weeks punching the max to competition. a function that is complete. a shippable component (doesn't mean to give it to the user but to sell the user dreams basically. (mid-level)
* User stories - program may have requirements which can be grouped into singularly large or bundle requirements. These can be broken down into an epic, story is a part of a sprint.

template: I want a microtransaction, how user logs in, how user receives invokes (low level) 2- 4 weeks till completion.

* Backlogs - the work that needs doing, all the ethics of the project, 4 user stores, 4weeks till completion, priorities set in order) user-stories, the back log is task that need to be done in order or priority. in a sprint you cannot change the priority but, in a backlog, you can amend user priorities or needs for software adjustment.

**-Explain alternative methods of documenting requirements**

requirements documentation

product - all the things for the project to do. should be neutral to be able to build in any programming language.

functional - features based on businesses and technology combined.

software - or a specification. the blueprint of the project, this is for the developers to use for meeting their accomplishments.

types of discussion - acquiring data can take up to a year.

* questions
* surveys
* focus groups.
* role play
* interviews
* brainstorming
* observing and analysis
* interface analysis