**Beet Seed**

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | ☺ | ☹ |
| **Waterfall** | 1. Consequent at execution. | 1. No room for maneuvers within the stages. |
| - medicine | 2. Well thought through. | 2. One stage may last unexpectedly long, not allowing to start working on the next step, postponing the deadline. |
| - space | 3. Attention to detail at every stage. | 3. Due to the field of usage, the responsibility and pressure are incomparably stronger. |
| - military | 4. More stable in terms of task distribution and stress. | 4. Is longer to design, develop, and test. Overall making the SDLC extend for multiple months/years. |
| - civil aviation etc. | 5. Is more precise and decreases the chances of mistakes after the product release. | 5. Is less "friendly" for additional changes and modifications since they may push the SDLC back to execute all the necessary tasks to add them. |
| - overall tendency: any field where malfunctions and mistakes might lead to disastrous consequences. | 6. Is fault preventive. | 6. Interpersonal communication has a lesser priority. It may lead to worse understanding and communication. |
|  | 7. The documents are well-maintained and composed. | 7. Fits less the teams with more independence and self-management. |
|  |  | 8. The end product might have more chances to be slightly or quite outdated when released. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  | ☺ | ☹ |
| **Kanban** | 1. Is very flexible. | 1. The SDLC may go back and forth. |
| - various software fields | 2. Allows to complete more tasks within a shorter time span. | 2. The documentation may not be that well maintained. |
|  | 3. The SDLC runs faster and shows the first results in a relatively shorter time period. | 3. Gives more room for potential bugs and additional tests to fix the same issue. |
|  | 4. Gives a better idea of how many tasks and how many of the requirements have been developed/implemented/tested. | 4. Does not fit the teams with low discipline, self-management skills, and autonomy. |
|  | 5. Enables the team to quickly fix, add, modify, develop, and test. | 5. Certain non-urgent tasks might be completed way later due to lower priority. |
|  | 6. The product is released faster and has much room for improvement even after its release. |  |
|  | 7. Can be improved way more with the users' reviews and feedback. |  |
|  | 8. Is more focused on the result and the team's productivity. |  |
|  | 9. People's communication is valued a lot, providing a better environment for clearer communication and less stress. |  |
|  | 10. Allows to prioritize tasks correspondingly to the goals. |  |

**Beet Sprout**

**Agile manifest**

With the development of technology, new inventions, and discoveries, the previous non-flexible methods showed to lose some of their effectiveness and productivity. The growing market, expanding services, and stronger competition required a new way of creating software products that could satisfy the customers' needs and win more audiences, thus enabling the product owners and their teams to keep their business on float and develop. As a result of the changes, new methods have been created.

However, knowing the agile methods does not guarantee that every person or team will automatically understand how to implement them in real life. Based on this realization, there is a need to have the Agile manifest explained and systemized. The main principles help better understand how, where, and when to use the methods that fall under the Agile umbrella.

On the other hand, modern values and society differ immensely from the one a decade or even a few years ago. Hence the previous way of coworking or communicating ideas might not fall within the values and objective reality. In a world where soft skills are more recognized than before, the old non-flexible methods may potentially bring more damage and deconstructive experiences.

When the main Agile manifest principles are clearly outlined, it is easier to choose one better fit for the team and the goals of the project/product. Consequently, the choice then helps to match the product owner's requirements and customers' expectations.

**Mighty Beet**

**Start-Up**

If I were to create a strut-up, that would be a platform and its mobile application for self-care, self-management, and self-coaching – “Self-a-Ware.”

*Economical scrutiny, lack of wealth, somewhat lacking education on the importance of mental health, pandemic, and various external/internal factors definitely do not help people live and enjoy their lives, let alone grow or achieve their goals. For this exact reason, a platform addressing these problems and providing free resources and channels to meet people going through the same might help others tackle their scrutiny.*

The platform would be a “collection” of free courses, lectures, or workshops in the fields of general healthcare, psychological support, financial literacy, professional orientation, time management, self-management, etc.

For such a complex compound start-up, it may be better to use a combination of Agile methods: the base principles would be the “*scrumban*” (srum+Kanban) with XP elements.

Since the idea of this start-up touches sensitive and “fragile” topics, people working on it would be those who are interested in sharing the experience, self-aware, and independent personas. They are able to distribute their workload and plan the working day to be productive but also not exhaust themselves to burnout. Hence, none of the classic methods, like Waterfall, V-Models, etc., would work here productively.

The sprints, like in scrum, but for longer periods, could help plan and complete the SDLC stages and meet the requirements. The backlogs would help keep track of the tasks, yet they would be also manipulated and “moved” the same way as a “Kanban board” would.

However, to delegate the tasks, there would be people responsible for watching the progress, those resolving upcoming problems (software malfunctions, technical issues, etc.). So, the team would have a clearer structure due to their roles and responsibilities as compared to the scrum method.

The XP method might take place for urgent, so-called “emergency” cases. For the times when every task might seem an “emergency,” I would suggest implementing the [Eisenhower Matrix](https://reclaim.ai/blog/eisenhower-matrix).