**Lesson 3 Cíntia Domingos**

**Beet Seed**

**1.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nº** | **Name of methodology** | **Pros** | **Cons** | **Industry** |
| 1 | Waterfall/cascade | - Review at the end of each phase: makes it possible to check if the project is on the right track and should continue or not  - Easy to use and manage: it’s a rigid model with specific deliverables and reviews for each phase  - Suitable for smaller projects with concrete requirements, because it’s easier to define everything clearly from the start | - Low flexibility: when the product reaches the testing phase, it’s not easy to go back to a previous phase to change something  - Product only available at the end of the lifecycle  - Not suitable for long projects with requirements that may change  - Needs more experienced resources  - Higher cost of fixing failures, because the project needs to be updated from the beginning | Government and defence programs: clearly defined project and requirements from the beginning, minimal changes expected |
| 2 | V-model | - Testing begins at the requirements gathering stage: this saves time and means a bigger chance of success comparatively to the Waterfall model  - Early testing also means defects are found early  - Each phase has its own round of testing  - Specific deliverables for each phase  - Thorough documentation for each phase | - Low flexibility  - No prototypes of the product, because it’s only developed at the implementation phase, which also means higher risk of an unsuitable product  - If any changes need to be made, the test and requirements documentation needs to be changed  - Needs more experienced resources | Software for space programs or medical equipment: complex systems with very clearly defined requirements and meticulous planning, high emphasis on testing |
| 3 | Iterative model | - Improvement when iterating over versions: the feedback received after releasing one version helps determining what should be kept, improved or discarded in the next one  - It’s possible to see the big picture from the beginning, which helps guiding the whole development process  - It’s easier to make changes as the product is constantly being improved, which also avoids the downflow of defects  - Possible to present each iteration to the stakeholders: they see that the project is evolving and can provide their feedback  - Makes it easier to deal with uncertainty  - Flexibility | - Costly unexpected issues may occur, because not all requirements were defined upfront  - May need additional resources | Videogames: companies frequently release alfa and beta versions of their games to receive feedback from players and improve the game in the next iteration. Even after full release they can continue upgrading the game with more features |
| 4 | Incremental model | - Solution built in parts: it’s easier to break big tasks into smaller tasks to concentrate on one task at the time, abstracting from the others  - Because the project is broken into smaller tasks, the list of requirements for each part are smaller and easier to manage  - No need to have the resources or the skills for the whole project at the beginning, because only a part of it is built at a time | - Does not reduce the risk of delivering an unsuitable product: the final product is only available at the end of the final stage of the process, you need to wait to review it, deliver it and get feedback  - Possible need to rework on a piece, which wastes time, energy and funds  - The definition of the complete product needs to be done at the beginning to determine the increments | Construction: each phase of the construction builds on the previous phase (foundations -> structure -> utilities -> finishing) |
| 5 | Spiral model | - Allows adjustments in the process and, consequently, improvements in the project, due to its iterative nature  - Enhanced risk avoidance, due to the increased risk analysis at each stage  - Prototype creation at each phase allows for checking the feasibility of the project  - Regular testing, which helps frequent detection and resolution of defects | - Time-consuming and expensive: requires lots of resources  - Challenging process due to the complexity of big projects  - Difficulty in time estimation because the number of stages is initially unknown | Operating system development: intricate and complex projects that evolve with changing hardware and requirements. The spiral model allows for continuous risk assessment and adaptation |

**Beet Sprout**

**2a)**

The manifesto appeared in the beginning of 2001, when 17 renowned developers, representatives of different software development methods, held two meetings to discuss the issues around these processes.

They were frustrated with the problems around traditional development processes of the 90s, which were a consequence of the huge increase in personal computing, leading to significant changes in the IT world, but not so much in the methodologies used.

As a consequence, the projects took too long to be developed, went over the budgets, had difficulty in adjusting to changes in the requirements and the communication between the development teams and the clients was insufficient, which meant the product might not always comply to the clients’ requirements. Besides this, there was a lot of emphasis on documentation and planning, which didn’t always reflect on the best results for the project.

The 17 developers came together to address these problems and try to find a way to make the development processes more flexible, efficient, adaptive and collaborative and thus appeared the Agile Manifesto, a set of values and guiding principles that aimed to change the way things worked.

The four values are as follows:

1- Individuals and interactions over processes and tools

2- Working software over comprehensive documentation

3- Customer collaboration over contract negotiation

4- Responding to change over following a plan

While the traditional methodologies, represented on the right, retain some of their value, the Manifesto places more value on the items on the left. “Individuals and interactions” acknowledges the importance of people, communication and collaboration during the development process. “Working software” means that delivering a solution, a working software, is more important than presenting extensive (and often excessive) documentation. The third value, “Customer collaboration”, emphasizes how important it is to collaborate with the customer during the project, so as to receive continuous feedback and adapt the product to the changes the customer may want. Last, but not least, “Responding to change” recognizes that changes in requirements, technology and business environment do happen and that the project needs to be able to adapt to them to be successful.

The 12 principles that accompany these values provide practical guidance to implement these practices, without being too rigid.

In short, the Agile Manifesto arose to challenge the *status quo* and to try and create a more flexible and collaborative approach to software development with the highest priority of making the customer happy. It has become a philosophy that guides development teams in a world of constant change.

**b)**

As mentioned, the Agile Manifesto was created to try to solve the problems of the traditional development processes, namely:

- Rigidity of the methods: previous methods were rigid and had a sequential approach, meaning it was not possible to adapt to changes in the requirements, technology or the market after starting to develop the project. Agile solves this problem by promoting a flexible and adaptive approach through iterative cycles and continuous feedback, allowing for changes to be accommodated.

- Speed: before Agile, the projects took too long to be completed because the focus was on extensive documentation. This meant that it was possible that the product was not as relevant to the market by the time it was released. With Agile, the focus is on the quick delivery of value, in the form of small, iterative releases, which means the product reaches the market much faster and it keeps getting improved.

- Collaboration with the customer: with previous methodologies, there would be an initial business plan for the project and little to no communication between the development team and the customer after that, which could result in products that did not meet all the needs of the customer. Agile prioritizes the continuous collaboration with the customer and the valuable feedback they provide to ensure that the products meet their expectations.

- Collaboration within the team: previously, there was little collaboration within the development team: the developers did their part, the QAs did theirs, and so on. The Agile methodology emphasizes the collaboration between every role in the team, which contributes to a more cohesive and open working environment and to a better and quicker resolution of issues.

For these reasons, the Agile methodologies have been embraced by companies all over the world, changing the work culture and increasing customer satisfaction.

**Mighty Beet**

**2)**

As the founder of a startup, my goals are to release a quality product in as little time as possible to start getting return on the investment quickly. In my opinion, an Agile methodology works for this and has many advantages.

The key principle of Agile is development in short iterations. This would allow for a rapid release of the product to the market while also allowing it to be continuously improved. Another advantage of this would be the feedback that could be received from early on, leading to improvements in the following iterations.

Another advantage is the flexibility of this method. Startup environments are dynamic and it’s important to be able to prioritize according to the needs, the changes in the market and new technologies, which Agile allows.

At the same time, this also works as a risk mitigation strategy, in the sense that the company doesn’t need to have all the funds and resources available from the start and can adapt to changing demands, reducing the risk of ending up with a product that doesn’t meet users’ expectations. The continuous testing processes also help address and fix any issues early on.

Lastly, Agile places a great emphasis on the collaboration with the stakeholders, which includes regular reviews of the product and real-time feedback. This is essential in conceiving an adequate product.

In short, Agile provides the flexibility, speed, risk mitigation, customer collaboration, continuous improvement and resource optimization that are essential to the needs of a startup.