**Beet Seed**

| **Methodology** | **Pros** | **Cons** | **For what industry is appropriate** |
| --- | --- | --- | --- |
| Waterfall | Clear and direct project requirements;  Easy management;  Its sequential structure allows easier planning and easier control of the project’s development. | When we are on a stage, we can’t go back to the previous ones, so it’s easier to mess up the project if you don’t catch the defects on that specific stage;  Little involvement of the client during the process;  Not flexible if we want to change or add new requirements during the project;  Testing usually cames at the end, on its set stage, so can lead to a late detection of errors. | Best for projects that have clear requirements, such as construction, manufacturing, IT and software development. |
| V-Model | Identifies flaws during the testing phase;  Guarantee of success using this model is higher;  Identification of defects can be done right at the beginning, not just at the end;  Simple and easy to use;  Better communication with the client. | More expensive;  Not good for complex projects;  Not flexible if we want to change or add new requirements;  Relies a lot on documentation;  More time consuming, as relies on a lot of documentation. | Automotive Software Development, as it was an early validation and testing. |
| Iterative Model | Easier to find functional and/or design defects at an early stage;  Less costly if we want to change the requirements at the middle of the process;  It supports changing the requirements if needed;  Issues and defects found on a specific iteration can be used to detect other ones on the next;  More involvement of the client as software is produced early and requires validation and feedback from the customer. | Not suitable for smaller projects;  When does the project end is uncertain, which can constitutes a risk;  Issues can appear because the requirements are not so well defined right at the start of the project. | Popular in technology, software development, design, qualitative research. |
| Incremental | Easier and less expensive to change requirements;  Errors are easier to identify;  Bigger involvement of the client, that can give feedback in each phase;  Easier to manage risk because risky issues are identified and handled during the iteraction. | Correcting a problem in one phase require correction in all the other fases, which can take longer;  Bigger need of good planning;  Needs clear and complete definitions of the whole system before starting working on it. | Software engineering. |
| Spiral | Software is produced early on the SDLC;  Flexibility when it comes to changing the requirements at a later phase;  Big involvement of the client and better communication between client and development team;  Better risk analysis and handling;  Great for large and complex projects. | Not good for smaller projects because it’s more expensive;  More complex;  Time estimation is more difficult because the number of phases is unknown;  Time consuming, as it requires many reviews. | Complex and big software engineering projects. |
| Agile | Flexibility in changing requirements if needed;  Environment of communication and collaboration between team members and clients; | As Agile relies a lot on the client’s feedback, it’s harder to define timelines to deliver the product; | Many such as aerospace industry, product development, healthcare, non-profits, advertising and marketing,… |

**Beet Sprout**

1. a) The Agile Manifesto appeared to respond to the limitations of the traditional software development methods, that involved a lot of inflexibility to receive new changes, bad communication between teams and between teams and client and an intense reliability on extensive documentation.

b) This Manifesto attempted to address these issues by promoting values such as individuals and interactions instead of processes and tools, working software instead of comprehensive documentation, customer collaboration instead of contract negotiation, and responding to change instead of following a plan.

In other and more simple words, these limitations involved being too rigid, not being good to receive new changes and not to involve the client on the process. So, this Manifesto appeared to solve this problems with better teamwork, better communication, much more flexibility and a not so strict reliability on documentation.

I guess it may worked for many projects and companies, but in the other hand, it can not be the best for some teams, I believe it depends on the company preferred methodology and approach.

But it was indeed widely implemented in this industry and is really well-known everywhere. And its advantages are undeniable.

**Mighty Beet**

2. I would choose Agile.

Since startups are smaller companies and are still starting their business, they probably won’t have that many employees. It’s important that the ones that work for the startup have good communication between each others. The Agile method supports that: an environment of collaboration, teamwork and open communication.

Also, in startups, time really matters, you want to develop your product faster so people get to know it faster too. In Agile, you deliver your work in smaller increments, one at a time, and the customer gives their feedback during all the process, so you know what’s good and what’s not so good and you can make the necessary adjustments on time.

Also, this method’s flexibility to change requirements is really good if you’re starting a startup. If you want to add some new things at the middle of the project, Agile method supports it.