Traditional approach

The oldest and most famous model of building a multilevel development process is the cascade (or simply waterfall) model: in it, each development stage corresponding to a stage of the software life cycle continues the previous one. That is, in order to move to a new stage, we must completely finish the current one.

The classic waterfall model consists of five steps:

- **Requirements gathering**. This is where the requirements for the project are gathered and formalised into a statement of work that outlines the work plan, anticipated risks and roles in the team;
- **Design**. This is where the main principles of the product are defined, e.g. software logic, building architecture, aircraft design. Tools are selected for these principles, e.g. programming languages, construction methods, aircraft modelling techniques;
- **Development**. This is where the product is made according to the plan and specification: code is written, the building is built, the aircraft is assembled. Development takes up most of the project;
- **Testing**. This is where the product is checked for compliance with the specification, errors are found and corrected;
- **Operation and Support**. This is where you release and maintain the product: fix bugs, keep it working, collect feedback from users, add new features.

The cascade model is simple and clear, but it is not as practical as it used to be. With dynamically changing requirements, a strictly structured process can turn from an advantage into a hindrance to the successful completion of system development. Therefore, today the waterfall model is mainly used by large companies for large and complex projects that involve comprehensive risk control.

Pros and cons of the cascade/waterfall model:

Pros

- Full documentation of each stage;
- Clear planning of deadlines and costs;
- Transparency of processes for the customer;

Cons:

- Necessity to approve the full scope of requirements to the system at the first stage;
- If it is necessary to make changes to the requirements later - return to the first stage and redo all the work done;
- Increased costs and time in case of necessity to change the requirements.

Agile approaches

Agile management methodology was invented to solve a number of problems of the classical/cascade/waterfall methodology. For example, too much emphasis on planning and the impact of delays in some teams on the work of others. This required a complete rethinking of the way project work is viewed, rather than changing any individual mechanics.

'Agility' is the ability of an agile team to constantly adapt to changing conditions and is achieved through three things:

Iterativity: Instead of taking a long, long time to come up with a plan and then taking even longer to make the perfect version of the product, an agile team tries to release a workable prototype as early as possible and then test and refine it time after time. Iterativity can have a negative impact on development time, but you have a more or less working product almost immediately.

Self-organisation : Everyone is equal in the team, there are no managers and supervisors, so there are no hellish co-ordinations. It saves resources, especially time.

Knowledge interpenetration : Any specialist in an agile team should have at least basic knowledge of related specialities. In addition to cross-functionality, constantly diving into new topics keeps your brain sharp.

Agile methodology is the values described in the Agile Manifesto or Agile principles:

- **People and interactions are more important than processes and tools.** If your team has principles, traditions, structures, tools, or conditions that are clearly interfering with the work - you should get rid of them. People should choose the way they organise themselves, the set of processes, the tools they use. In the end, it should all help the work, not hinder it.
- **A working product is more important than documentation**. This doesn't mean 'working Agile means working without documentation'. Agile teams also have documentation, but they don't spend a huge amount of time and resources on it.
- **Collaboration with the customer is more important than agreeing on contract terms.** Look a little further than agreeing on technical specifications and estimates. There is no point in spoiling the relationship with the customer, even at the cost of timely payment. If you can't agree on the work and spoil communication, you'll end up losing that customer and possibly the next ones. Any contracts, documents and agreements should benefit your customer relationships, not damage them
- **Willingness to change is more important than sticking to the original plan.** Even if you have a project plan, it will almost certainly have to be changed over time - this is the essence of Agile.

Pros and cons of Agile

Pros

- Adaptability to change
- High team engagement
- Helps meet deadlines
- Helps you make the product you really need

Cons

- It's all about the team and motivation
- No clear plan
- Can't quickly change from inflexible to flexible methodologies - and back again

Summary:
The **waterfall model** offers structured, well-documented processes ideal for large, complex projects but lacks flexibility, leading to increased costs and delays when changes are needed. Conversely, the **Agile methodology** promotes adaptability, continuous improvement, and strong team collaboration, making it better suited for dynamic environments. However, Agile relies heavily on team motivation and lacks a clear, fixed plan, which can be challenging for some projects.

Both methods have significant advantages and disadvantages. Which one is best in a given situation depends mainly on the attitude of the customer, the funding model, the scale and the time horizon of implementation.