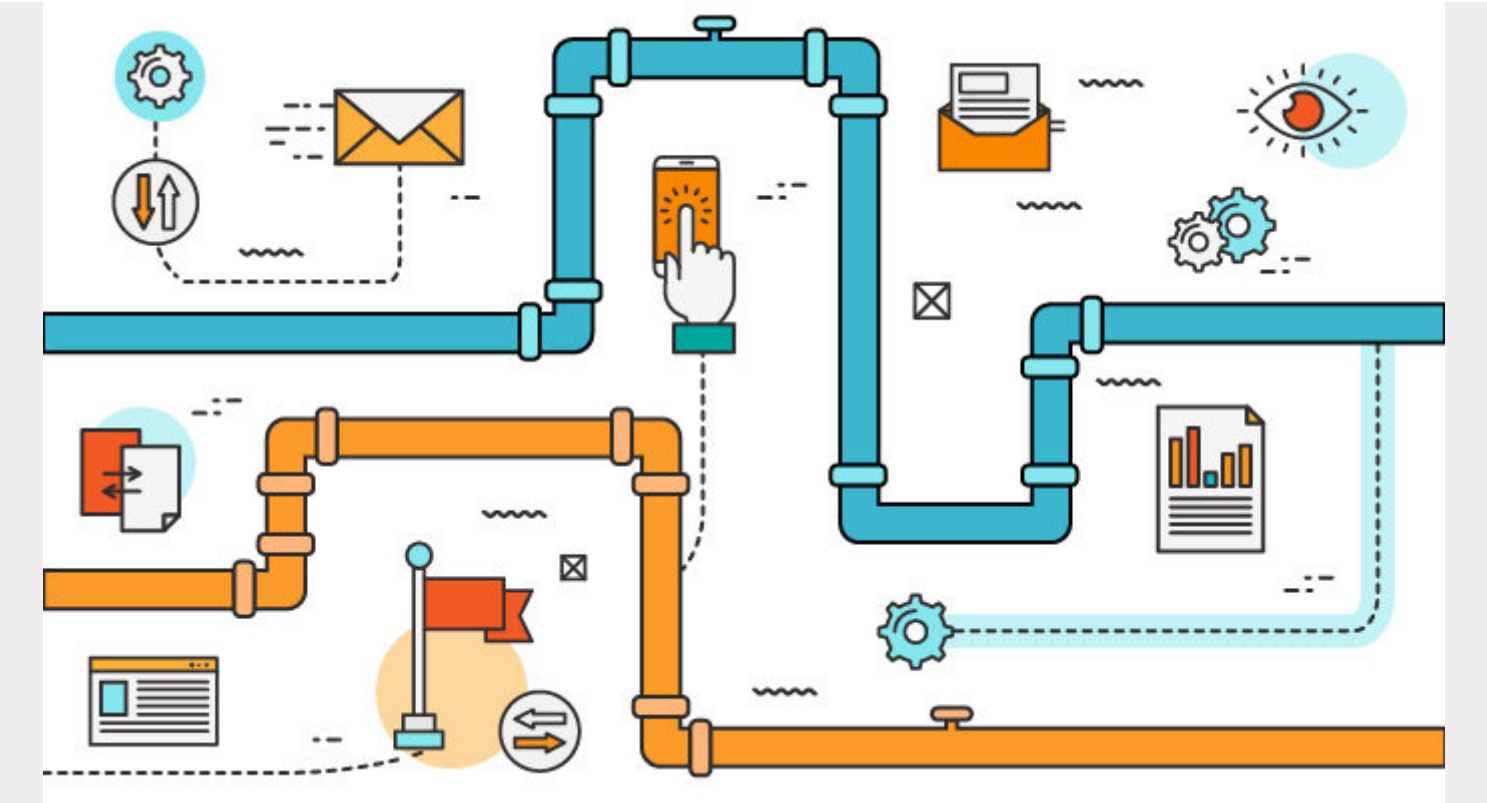
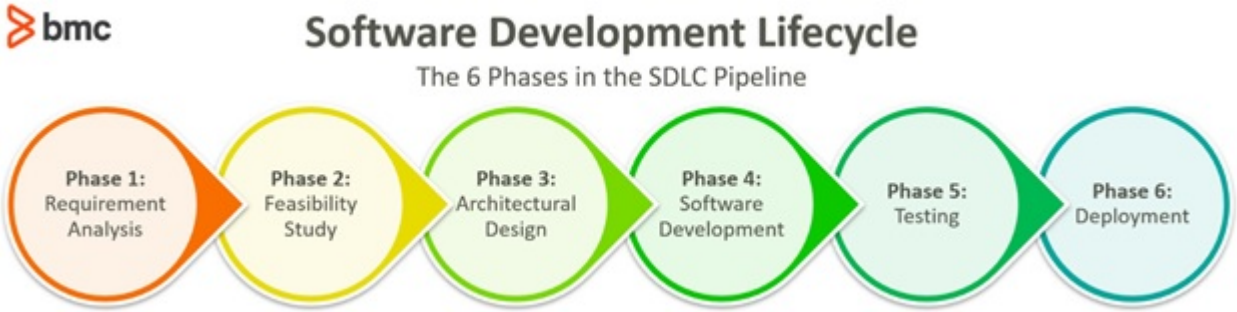
# AGILE VS WATERFALL SDLCS: WHAT'S THE DIFFERENCE?



Agile and Waterfall are both Software Development Lifecycle (SDLC) methodologies that have been widely adopted in the IT industry.

The Waterfall framework was designed to enable a structured and deliberate process for developing high quality information systems within project scope. The spirit of becoming more adaptive through the real-world implementation of a software project plan gave way to the Agile methodology.

Both Waterfall and Agile require organizations to follow certain operating principles—but the practice often departs from the principles. It is therefore important to understand what Agile and Waterfall mean and how they differ as you make your choice for an SDLC framework that best suits your development goals.



## Waterfall principles

Waterfall is a classical SDLC methodology that follows logical progression of linear and sequential phases within the project lifecycle process. Some of the key principles in Waterfall include:

- **Sequential structure.** Typically includes phases such as Requirements, Design, Implementation, Verification, and Maintenance.
- **Strong focus on documentation.** Extensive details that define project requirements and implementation process.
- **Low customer involvement.** Requirements must be agreed early during the project lifecycle. Once requirements are defined, the development process is strongly focused on meeting the agreed requirements.

## Agile principles

The Agile SDLC model is designed to facilitate change and eliminate waste processes (similar to Lean). It replaces a command-and-control style of Waterfall development with an approach that prepares for and welcomes changes.

The key differentiating Agile principles include:

- **Individuals and interactions over process and tools.** Build a strong communication and collaboration mindset between team members across all functional domains.
- **Working software instead of comprehensive documentation.** Tactic knowledge is more valuable than document knowledge that's often difficult to communicate and never truly sufficiently descriptive. Communications that take place while working together are likely to effectively communicate the necessary information between team members—as opposed to writing this information in a large documentation resource.
- **Customer collaboration over contract negotiation.** Requirements change rapidly, especially when projects take prolonged duration to build and deploy, whereas as the market dynamics changes unpredictably.
- **Responding to change over following a plan.** A successful project development model that has the provision to adapt can help the vendor and customer meet collective goals without exceeding project scope.

The 12 principles behind the Agile Manifesto are detailed here.

## Waterfall vs Agile: Key Differences

bmc

| Agile | Waterfall |
| --- | --- |
| Iterative development in short sprints | Sequential development process in pre-defined phases |
| Flexible and adaptive methodology | The process is documented and follows the fixed structure and requirements agreed in the beginning of the process |
| Feedback-based approach: Sprints lead to short build updates that are evaluated on and guide the future direction of the development process. | Limited and delayed feedback: The software quality and requirements fulfilment isn't evaluated until the final phase of the development processes when testers and customer feedback is requested. |
| A provision for adaptability: Project development requirements and scope is expected to change over the course of the iterative development process. | The requirements and scope are definitive once agreed upon. |
| The SDLC phases overlap and begin early in the SDLC: planning, requirements, designing, developing, testing, and maintenance. | The SDLC phases are followed in order, with no overlap. Members of one functional group are not involved in another phase that doesn't belong to their job responsibilities. |
| Follows a mindset of collaboration and communication. The requirements, challenges, progress, and changes are discussed between all stakeholders on a continuous basis. | Follows a project-focused mindset with the aim of fully completing the SDLC process. |
| Responsibilities and hierarchical structure can be interchangeable between team members. | Fixed individual responsibilities, particularly in management positions. |
| All team members focused on end-to-end completion (achieved sequentially) for the projects. | Team members focused on their responsibilities only during their respective SDLC phases. |
| Suitable for short projects in high-risk situations. | Suitable for straightforward projects in predictable circumstances. |
| Limited dependencies as the focus is less on implementation specifics, and more toward the mindset. | Strict dependencies in technologies, processes, projects and people. |

# Issues with the Waterfall approach

The comparison between basic principles of Waterfall and Agile methodologies point to some key issues with the Waterfall model, especially when considering that it remained the de facto SDLC standard for decades:

- **The lack of communication** introduces highly varied interpretation of requirements and the documentation between team members.
- **The lack of adaptability** renders the model unsuitable for most consumer-focused software projects where the requirements must follow unpredictable, dynamic, and rapidly evolving external change agents.
- **The strict focus on fulfilling original requirements** discourages mistakes and change. As a result, creativity, innovation, and novelty are suppressed. Lack of flexibility prevents experiences that help identify areas of improvement or change for the better.
- **The lack of communication** between engineering teams, customers, and real-world users creates the lack of technical empathy. As a result, many expensive innovations fail to gain market traction, whereas agile startup firms focusing on specific customer problems gain popularity rapidly in comparison.

# Issues with Agile

At the same time, the Agile SDLC methodology hasn't proven to be a silver bullet. Although the principles of the Agile model aim to solve problems that may arise from the Waterfall approach, many organizations fail to realize the promised advantages. This issue emerges due to the following reasons:

- **Organizations follow Agile for planning**, but swiftly transition to "fast waterfall" sprints in implementation. The agility isn't truly designed and embedded during the execution.
- **There's no coherent system design** with practical provisions for improving or changing software functionality. As a result, iterating on customer feedback only adds to complexity. The organization is forced to follow the traditional Waterfall model as it fixes each issue based on feedback.
- **The Agile mindset isn't truly followed across the organization**. Once the team meetings end, individuals may struggle to adapt—unless a system of change is introduced. For instance, siloed organizational departments and lack of collaboration tools restrict the ability to be truly collaborative and Agile in practice.
- **Technical debt accumulates fast** during the Agile process, especially if the sprints are solely focused on bringing new functional improvements instead of fixing quality issues early during the SDLC process.
- **Evaluating the wrong metrics** for project progress gives a false picture of success. Critical performance lapses aren't discovered until too late into the SDLC pipeline.
- **Following Agile isn't entirely sufficient**. The value-driven perspective requires organizations to identify the hurdles that prevent them from achieving the goals of the Agile SDLC methodology: fast-paced delivery of high-quality software, lower waste process, and satisfied customers.

## DevOps as the cure-all?

With the aim of addressing these challenges, many organizations are now following the DevOps SDLC methodology that takes the Agile principles of fast and iterative software production, but inherently focuses on collaborative, continuous, and automation-driven processes in software development, testing, deployment, and delivery.

## Additional resources

For more on this topic, explore the BMC DevOps Blog and these articles:

- The Role of Agile in DevOps
- DevOps vs Agile: A Complete Introduction
- Accelerating Agile Delivery with ITSM Integration
- Managing IT as a Product—Not a Project
- Role of the Project Management Office (PMO) in an IT Organization