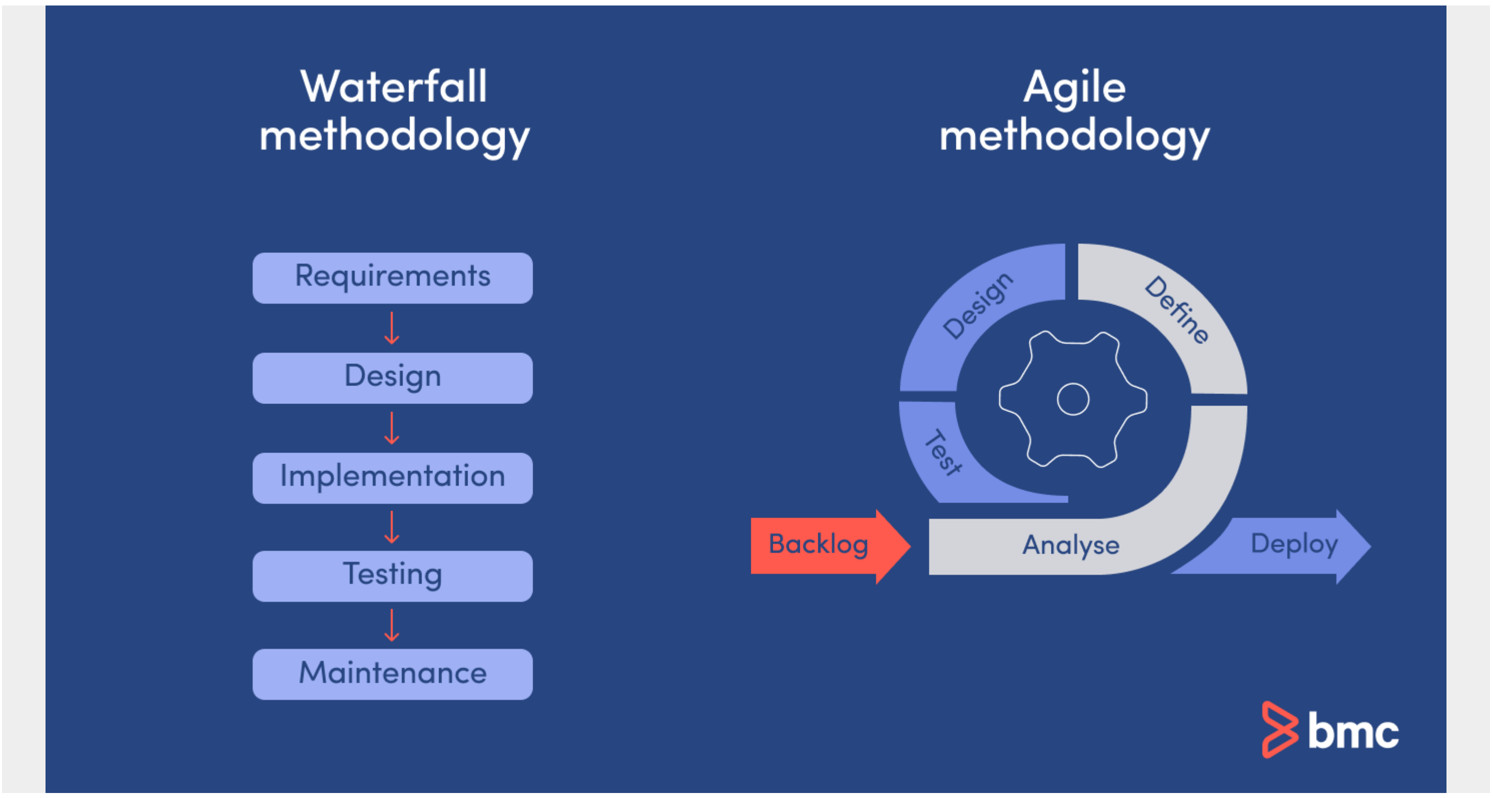
AGILE VS. WATERFALL: WHAT'S THE DIFFERENCE?

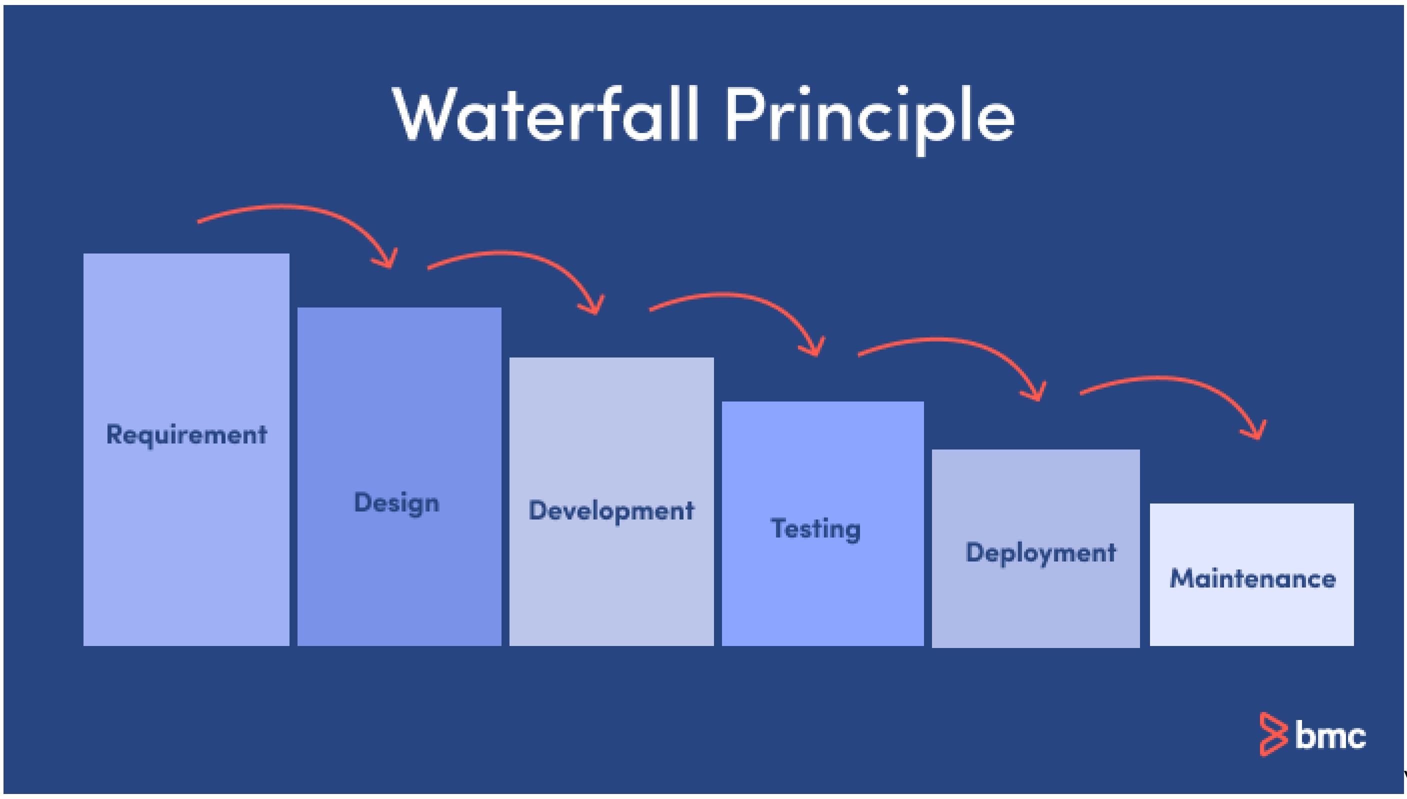


Agile and Waterfall are both Software Development Lifecycle (SDLC) methodologies that have been widely adopted in the IT industry.

The Waterfall framework was designed to enable a structured and deliberate process for developing high-quality information systems within the 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 software development lifecycles require organizations to follow certain operating principles, but practice often departs from these principles. It is important to understand what Agile and Waterfall methodologies mean and how they differ as you choose an SDLC framework that best suits your development goals.

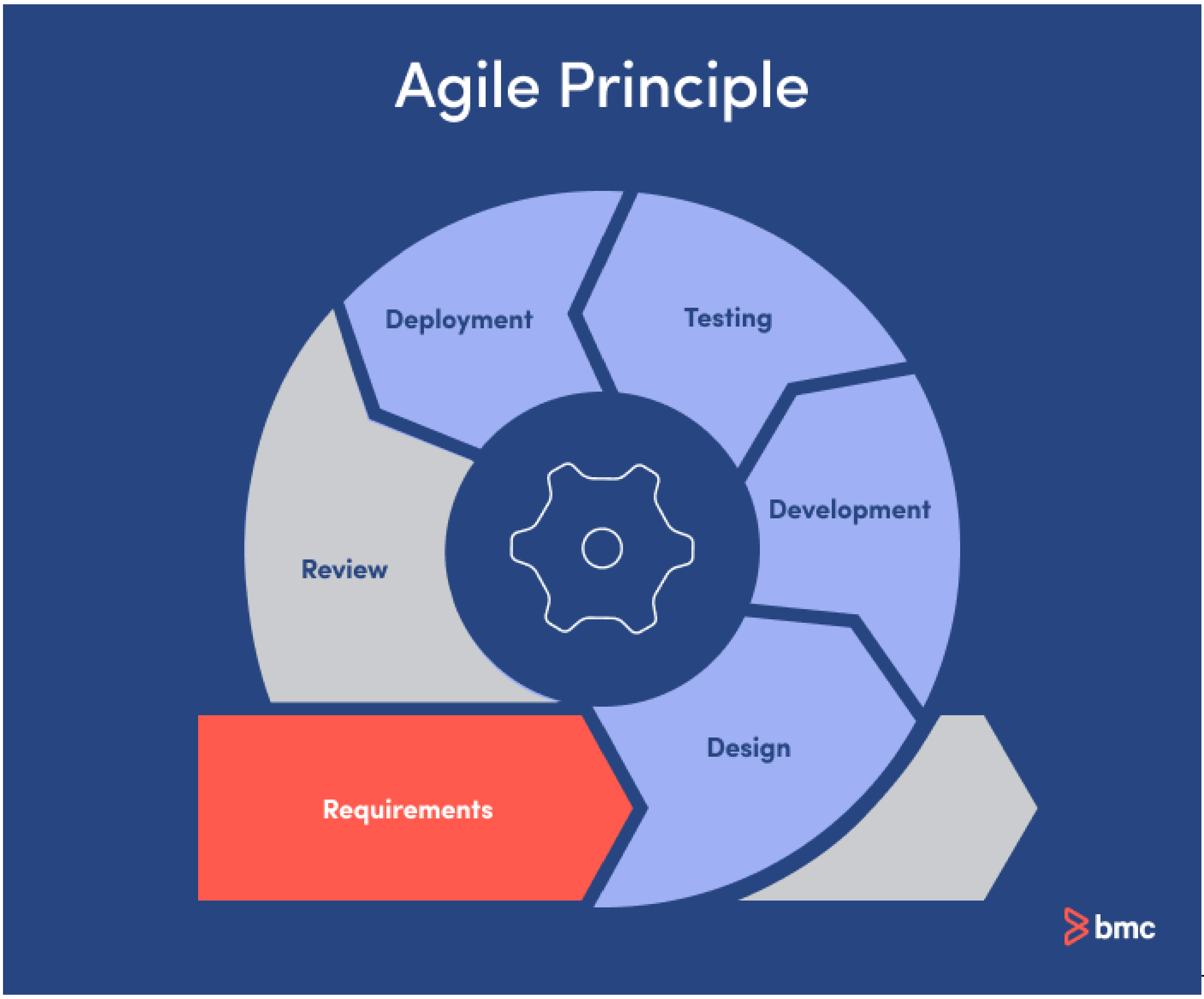
Waterfall principles



Waterfall is a classical SDLC methodology that follows the 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 upon early in the project lifecycle. Once 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, all of whom have certain roles and responsibilities in Agile.
- **Working software instead of comprehensive documentation.** Tactic knowledge is more valuable than document knowledge, which is often difficult to communicate and never truly sufficiently descriptive. Communication while working together is likely to effectively communicate the necessary information between team members instead of writing this information in a large documentation resource.
- **Customer collaboration over contract negotiation.** Requirements change rapidly, especially when projects take a prolonged duration to build and deploy, and the market dynamics change 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.

Agile vs. Waterfall methodologies: key differences

Drawbacks of Waterfall model

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

- **Varied interpretation.** The lack of communication introduces highly varied interpretations of requirements and documentation between team members.
- **Lack of adaptability.** 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.
- **Lack of flexibility.** 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.
- **Lack of technical empathy.** The lack of communication between engineering teams, customers, and real-world users creates a 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.

Drawbacks of Agile model

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:

- **Agility isn't truly designed and embedded during execution.** Organizations follow Agile for planning but swiftly transition to "fast waterfall" sprints in implementation.
- **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 a 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 focus 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.

Agile vs. Waterfall pros and cons

The right SDLC methodology depends on your project.

Agile model brings the flexibility of ongoing and continual changes during development. Teams work closely with the customer to fine-tune the product to meet their needs exactly. Work is iterative and done in short springs for faster testing and feedback. Because releases are frequent and adjustments are made just as frequently, the risk of delivering code that doesn't address a user's needs is lower. Agile teams are empowered to take responsibility and to innovate freely.

The Waterfall software development approach also has its strengths. It provides a clear structure of milestones and phases, giving projects a clear and predictable timeline and scope. Thorough documentation is also easier.

Considering pros and cons is the concept of the Waterfall drop and Agile rapids. In the Waterfall methodology vs. Agile, you run the risk of falling off the cliff. If your project splash lands and splinters into pieces, you are in serious trouble. With the Agile methodology, you bring the project through smaller drops with plenty of opportunities to avoid disaster and maneuver around hazards.

Agile and Waterfall project management

SDLC and project management are closely related in several ways, all of which have implications for Agile and Waterfall software methodologies.

- **Planning:** Project management involves defining objectives, laying out scope, and calculating project timelines. SDLC is the structure or framework for implementing the project plan.
- **Risk Management:** Project management involves identifying risks and planning how to deal with them. Risk management is integrated into each SDLC phase.
- **Control:** Project management is where SDLC work is monitored to keep the project on track.
- **Quality assurance:** Project management is about scheduling testing in the SDLC process.
- **Communication:** The role of project management is to ensure stakeholders are working well together in the SDLC process.

Learn from an expert BMC application developer about the practical benefits of an Agile approach to projects and the risks of Waterfall methodology.

DevOps as the cure-all?

To address these challenges, many organizations are now following the DevOps SDLC methodology, which adopts 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