# Agile & Waterfall Methodologies

# Agenda

- Introduction
- Agile Methodology
- Waterfall Methodology

# SDLC

- The **Software Development Life Cycle (SDLC)** is a process used by software developers to design, develop, test, and deploy software applications.
- It provides a structured approach for managing and guiding software projects from start to finish.
- **SDLC Methodologies: Agile vs. Waterfall**
  - While the SDLC framework defines the stages of software development, the **Agile** and **Waterfall** methodologies are two different approaches used within it.

# AGILE METHODOLOGY

- Agile methodology is an approach to software development that emphasizes flexibility, collaboration, and customer-centricity. It focuses on delivering small, working pieces of a product incrementally through iterative cycles.
- Agile methodology is a project management and software development approach that emphasizes **flexibility, collaboration, and customer-centricity**.
- It involves breaking the project into smaller, manageable parts and delivering them incrementally.
- This allows teams to adapt to changes quickly and provide customer value faster.

- **Key Benefits of Agile:**
  - Faster time to market
  - Increased customer satisfaction
  - Continuous improvement
  - Adaptability to change

# AGILE METHODOLOGY OVERVIEW

- Agile abandons the risk of spending months or years on a process that ultimately fails because of some small mistake in an early phase. The methodology relies instead on trusting employees and teams to work directly with customers to understand the goals and provide solutions in a **fast and incremental way**.
- **Faster, smaller:** Traditional software development relied on phases like outlining the requirements, planning, design, building, testing, and delivery. By contrast, agile methodology looks to deploy the first increment in a couple weeks and the entire piece of software in a couple months.
- **Communication:** Agile teams within the business work together daily at every stage of the project through face-to-face meetings. This collaboration and communication ensure the process stays on track even as conditions change.
- **Feedback:** Rather than waiting until the delivery phase to gauge success, teams leveraging an agile methodology track the success and speed of the development process regularly. Velocity is measured after the delivery of each increment.
- **Trust:** Agile teams and employees are self-organizing. Rather than following a manifesto of rules from management *intended* to produce the desired result, they understand the goals and create their own path to reach them.
- **Adjust:** Participants tune and adjust the process continually, following the ***Keep It Simple*** *(KIS) principle*.

# Key principles and values

- The Agile Manifesto, created in 2001 by 17 software developers, emphasizes the following **four core values**:
- **Individuals and interactions over processes and tools**
  - This value highlights the importance of people and communication in the development process. While tools and processes are necessary, they should not hinder collaboration or the exchange of ideas.
- **Working software over comprehensive documentation**
  - Agile focuses on delivering functional software rather than getting bogged down by extensive documentation. While documentation is useful, it should not be the primary focus if it delays the delivery of a working product.
- **Customer collaboration over contract negotiation**
  - Agile encourages ongoing communication with customers or stakeholders. The goal is to understand their needs and adapt to changes, rather than sticking rigidly to a predefined contract or set of requirements.
- **Responding to change over following a plan**
  - Agile values flexibility and adaptability. It acknowledges that requirements may change during a project, and the ability to respond to those changes is more important than rigidly sticking to an initial plan.

# Key principles & values

- The **12 Principles** of Agile:
- These values are supported by twelve principles that guide teams in implementing Agile practices effectively. Some of the key principles include:
- **Customer satisfaction through early and continuous delivery** of valuable software.
- **Welcome changing requirements** even late in development, ensuring the team can adapt.
- **Deliver working software frequently**, with a preference for shorter timescales.
- **Collaborate closely** with stakeholders and business representatives.
- **Simplicity** – the art of maximizing the amount of work not done.
- **Self-organizing teams** that make decisions to improve their productivity.
- **Regular reflection** on how to become more effective, followed by adjustments.
- Together, the **values and principles** of the Agile Manifesto help software development teams stay focused on people, adaptability, and delivering value to customers. They promote a flexible, iterative approach that values collaboration and ongoing improvement.

# Pros

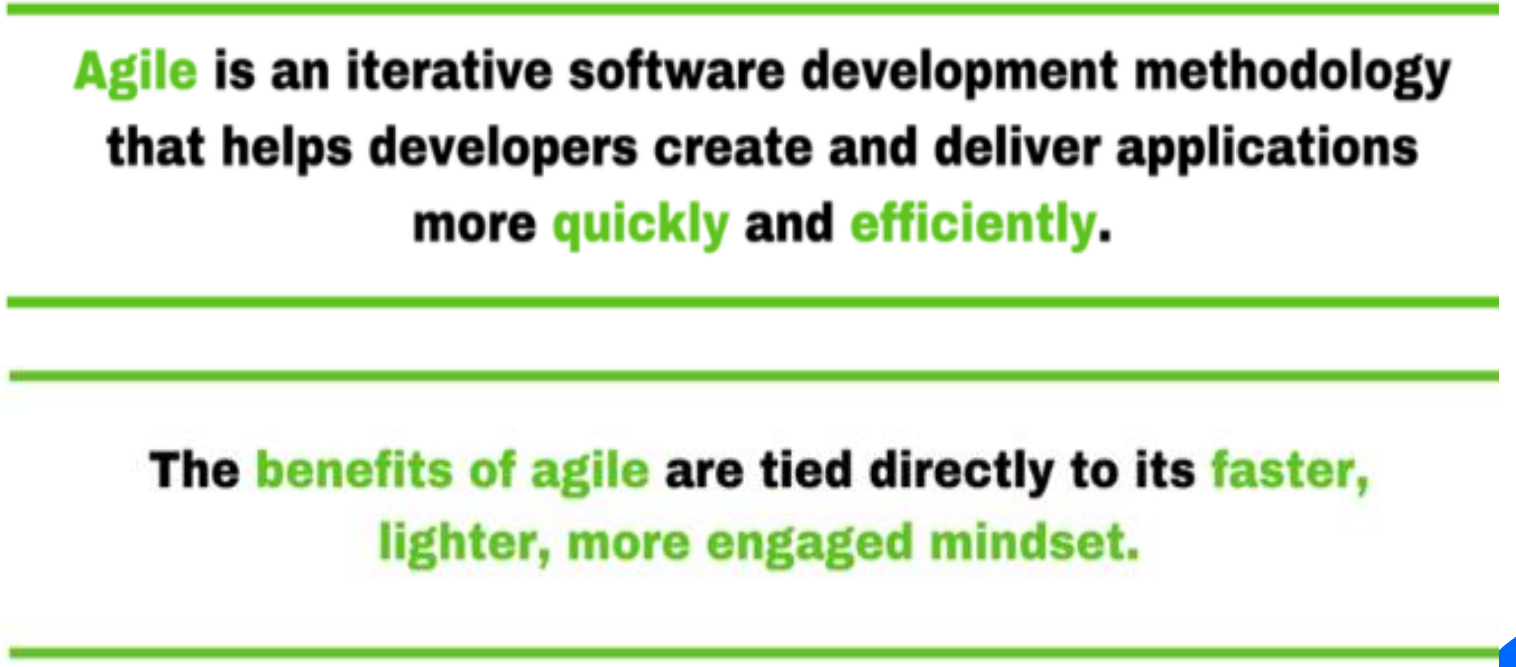
- **Flexibility and Adaptability**
  - Agile allows teams to easily adapt to changes in project requirements, even late in development. This makes it well-suited for projects with evolving needs or uncertainty.
- **Faster Delivery**
  - Agile emphasizes delivering smaller, functional software increments frequently, which leads to quicker releases and faster time to market.
- **Increased Collaboration**
  - The methodology promotes strong communication and collaboration between developers, stakeholders, and customers, leading to better alignment with customer needs.
- **Improved Customer Satisfaction**
  - Continuous customer feedback during each iteration ensures that the final product better meets their expectations and needs.
- **Higher Product Quality**
  - Regular testing and feedback cycles in Agile ensure that defects are caught early, improving the overall quality of the product.
- **Empowered Teams**
  - Agile relies on self-organizing, cross-functional teams, which can increase motivation and responsibility among team members, fostering innovation and accountability.

# Cons

- **Scope Creep**
  - Due to its flexibility, Agile can lead to scope creep if requirements are continuously changed or added without proper management.
- **Requires Strong Collaboration**
  - Agile relies heavily on communication and close collaboration. If there is a lack of effective communication, the process can become inefficient.
- **Can Be Resource Intensive**
  - Agile demands frequent meetings (e.g., daily stand-ups, sprint reviews), which can be time-consuming and require significant resources.
- **Not Ideal for All Projects**
  - Agile may not be suitable for projects with fixed requirements or those that require strict, upfront planning and predictability (e.g., large-scale infrastructure projects).
- **Less Documentation**
  - Agile prioritizes working software over comprehensive documentation. This can sometimes lead to a lack of detailed records, which may become an issue in future phases or handovers.
- **Difficulty Scaling**
  - While Agile works well for small to medium-sized teams, scaling it for large, distributed teams can be challenging and may require additional frameworks like SAFe or LeSS.

# 3 C's of Agile

- Agile is an iterative software development methodology that helps developers create and deliver applications more quickly and efficiently.
- It's based on the principles of collaboration, customer feedback, and the "three C's"—**card**, **conversation**, and **confirmation**.



# Agile Methodology Tools

- **Stackify Retrace:** For a more robust solution with monitoring, errors, logs, and more, Stackify's Retrace provides app performance insights from integration to QA to production, at the code level.
- **ActiveCollab:** An affordable tool for small businesses, ActiveCollab is easy to use. This software development aid requires little training and provides excellent support.
- **Agilo for Scrum:** Stakeholders get updated automatically on the project's progress with Agilo for Scrum. Features sprint reports and burn down charts for better data mining.
- **Atlassian Jira + Agile:** This powerful project management tool facilitates development by incorporating scrum, kanban, and customizable workflows.
- **Pivotal Tracker:** This methodology tool is geared specifically for mobile projects. A little jargon-heavy, it's user-friendly after a brief orientation period.

# Waterfall Methodology

- Waterfall methodology is a well-established project management workflow. Like a waterfall, each process phase cascades downward sequentially through five stages (requirements, design, implementation, verification, and maintenance).
- Waterfall is a traditional, linear approach to project management and software development where each phase is completed sequentially. The process follows a structured, step-by-step approach, with each stage being dependent on the completion of the previous one.
- Unlike other methods, such as the Agile methodology, Waterfall doesn't allow flexibility. You must finish one phase before beginning the next. Your team can’t move forward until they resolve any problems. Moreover, as our introduction to project management guide outlines, your team can’t address bugs or technical debt if it’s already moved on to the next project phase.

# Key Phases of Waterfall Methodology

- **Requirements Gathering:**
  - All project requirements are gathered and documented before the development begins. No changes are made once the project moves into the next phase.
- **System Design:**
  - Based on the requirements, the system design is created, including architecture, components, and overall design specifications.
- **Implementation (Coding):**
  - Developers write the code based on the system design specifications. The coding phase is separate from the design phase.
- **Integration and Testing:**
  - The developed product is tested for defects and issues. Integration testing ensures that all components work together.
- **Deployment:**
  - The final product is deployed for the end-users, typically after all bugs are resolved and testing is complete.
- **Maintenance:**
  - The system enters the maintenance phase, where ongoing support and updates are provided as needed.

# Pros of Waterfall Methodology

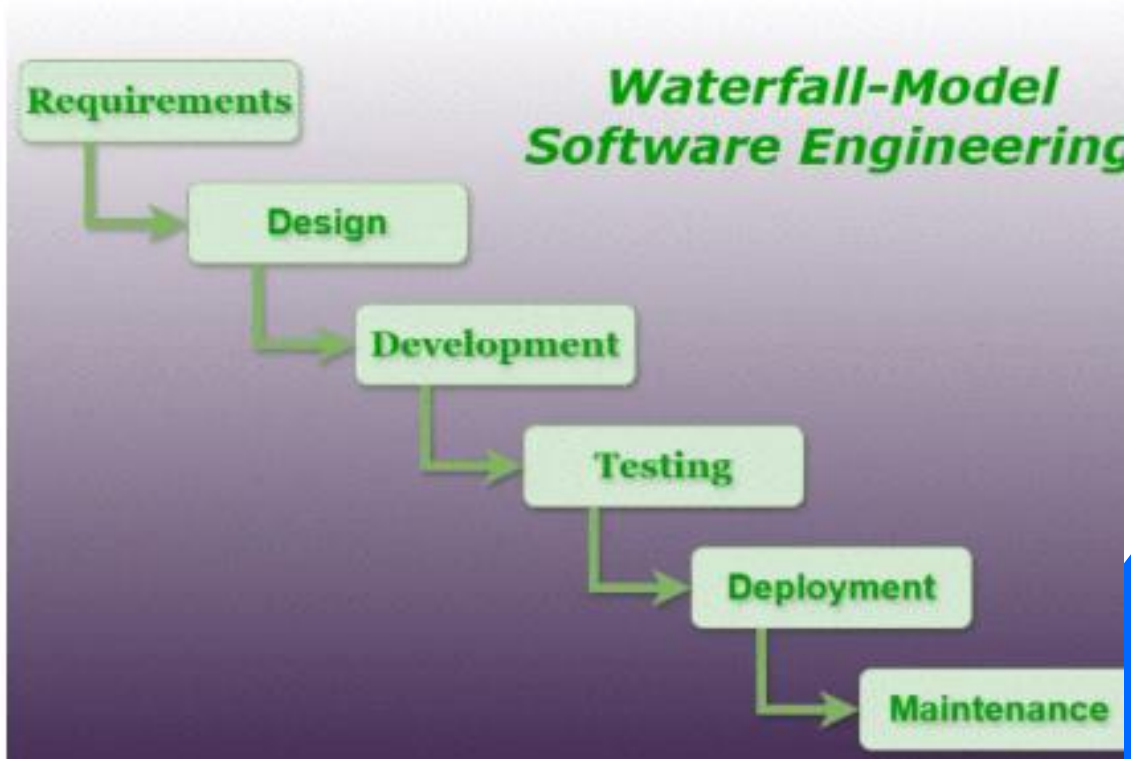
- **Clear Structure and Phases**
  - The sequential nature of Waterfall allows for clear project milestones and predictable timelines.
- **Well-Defined Requirements**
  - Waterfall is ideal for projects where requirements are clear from the start and are unlikely to change.
- **Easy to Manage**
  - Due to its rigid structure, Waterfall is often easier to manage, particularly for large, complex projects.
- **Documentation Focus**
  - Waterfall emphasizes documentation at each stage, ensuring that each phase is well-documented for future reference or maintenance.

# Cons of Waterfall Methodology

- **Inflexibility to Change**
  - Once the project moves past the requirements phase, making changes can be difficult and costly. This makes Waterfall less ideal for projects with evolving or unclear requirements.
- **Delayed Feedback**
  - Since testing happens at the end, issues and bugs may not be identified until later stages, resulting in costly fixes and delays.
- **Longer Time to Market**
  - The linear approach can delay the delivery of functional products, as everything must be completed in sequence.
- **Risk of Misalignment with Customer Needs**
  - If customer needs change during development, Waterfall's rigid structure may not allow for adjustments, which can result in the final product being less relevant to the end users.
- **Requires Extensive Documentation**
  - The need for detailed documentation at each stage can be time-consuming and may be seen as excessive for some projects.

# When to use Waterfall Methodology

- **Clear and Stable Requirements:**
  - Ideal for projects with fixed, well-understood requirements (e.g., regulatory compliance, infrastructure projects).
- **Large Teams and Complex Projects:**
  - Suited for large-scale, complex projects where strict management, clear milestones, and extensive documentation are necessary.



# Key Differences

| Aspect | Agile | Waterfall |
|---|---|---|
| **Approach** | Iterative and incremental | Linear and sequential |
| **Flexibility** | Highly flexible, allows changes at any stage | Rigid, changes are difficult after the requirements phase |
| **Project Phases** | Phases overlap and iterate through continuous cycles | Phases are distinct, each phase completed before the next begins |
| **Customer Involvement** | Continuous collaboration and feedback throughout | Limited customer involvement until the final delivery |
| **Documentation** | Emphasizes working software over documentation | Focuses heavily on documentation at each stage |
| **Delivery Timeline** | Frequent, small deliveries (sprints or iterations) | Single delivery at the end of the project |
| **Team Structure** | Cross-functional, self-organizing teams | Specialized roles for each phase (e.g., designers, developers, testers) |
| **Risk Management** | Early identification of issues through feedback | Risks are discovered late in the project, often during testing |
| **Suitability** | Ideal for dynamic, complex projects with changing requirements | Suitable for projects with clear, fixed requirements |
| **Time to Market** | Faster, with continuous releases | Slower, as all phases must be completed before release |

# When to use Agile & Waterfall Methodologies

- **Use Agile** if:
  - You have changing or unclear requirements.
  - Speed and flexibility are key.
  - You need frequent customer feedback and early releases.
  - You want continuous improvements and adaptability throughout the project.
- **Use Waterfall** if:
  - Requirements are well-defined, fixed, and unlikely to change.
  - The project needs strict documentation and compliance.
  - You can predict all project details upfront.
  - It's a large-scale, complex project with less room for iteration.

# Thank you