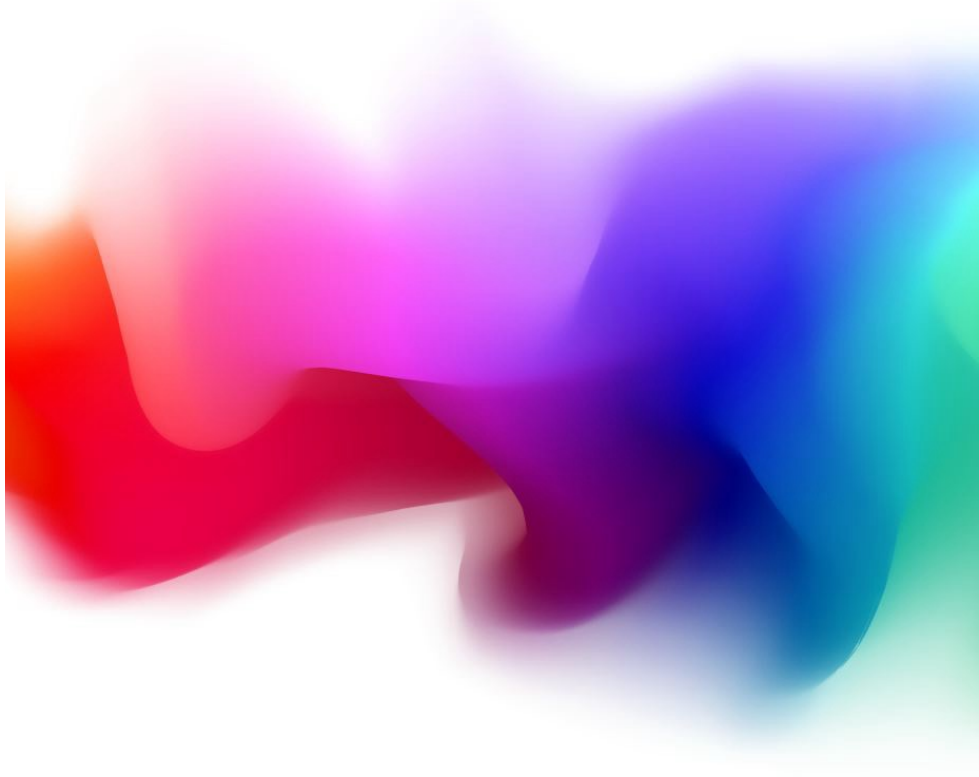
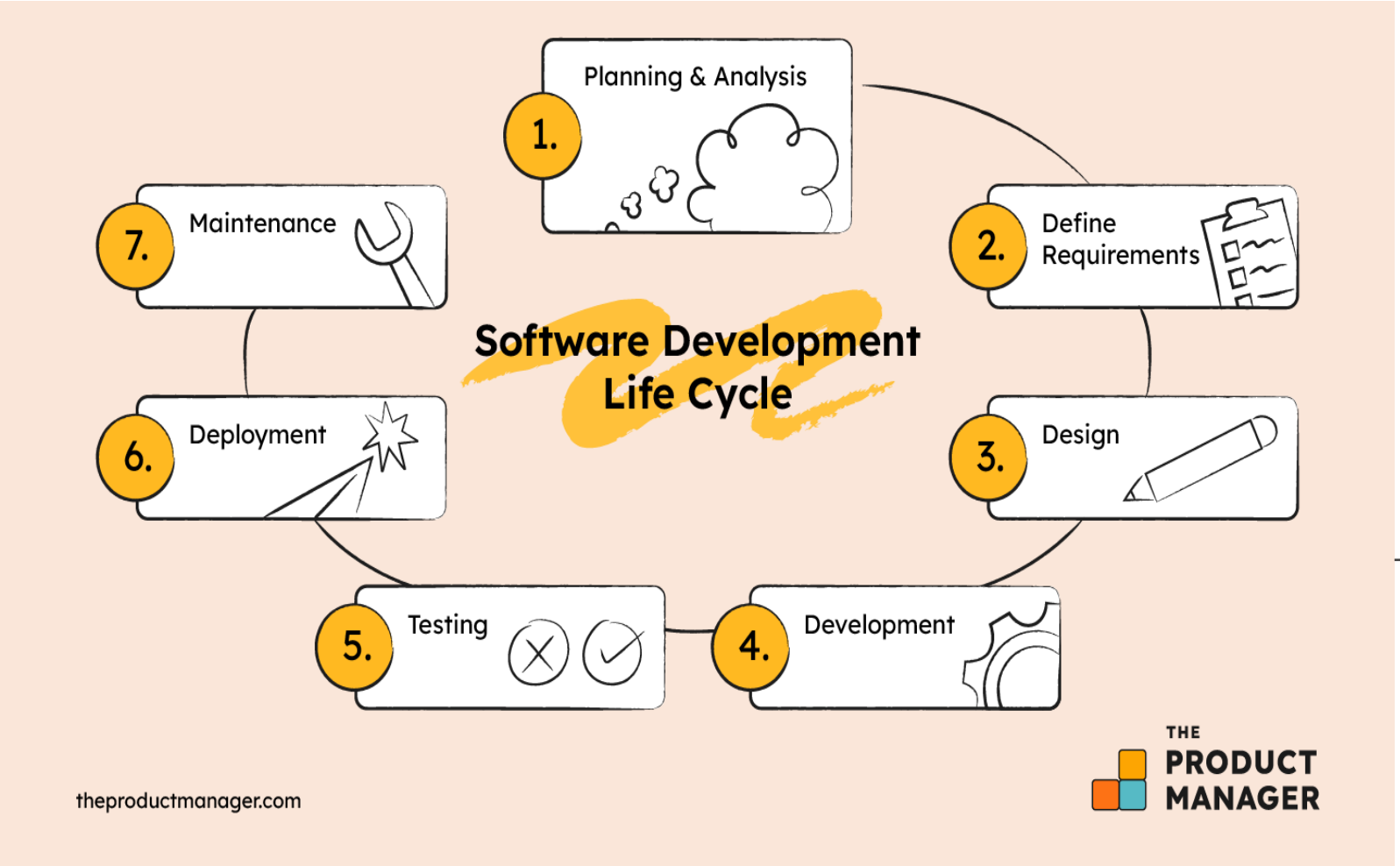
Development

IN-HOUSE, OFF-THE-SHELF, CLOUD





Software Development Life Cycle (SDLC)

1. Planning

Planning is the initial phase of the SDLC where the project's objectives, scope, purpose, and feasibility are determined.

This phase involves gathering high-level requirements and creating a project plan that outlines the timeline, resources, and budget.

Effective planning helps ensure that the project is aligned with business goals and sets the foundation for successful development.

2. Analysis

In the Analysis phase, detailed requirements are gathered and documented.

This involves understanding the business needs, identifying user requirements, and defining functional and non-functional specifications.

Analysts work closely with stakeholders to ensure all requirements are clearly understood and accurately captured.

This phase also includes feasibility studies and risk assessments to identify potential challenges and solutions.

3. Design

The Design phase translates the requirements from the Analysis phase into a blueprint for constructing the software.

This includes designing the system architecture, user interfaces, databases, and other system components. Detailed design specifications are created to guide developers in building the software.

The goal is to create a clear, detailed plan that addresses all aspects of the system, ensuring it meets user needs and performs efficiently.

4. Implementation

During the Implementation phase, the actual code for the software is written based on the design specifications.

Developers use programming languages and tools to build the software components, integrating them into a cohesive system.

This phase involves coding, unit testing, and integration of various modules.

The implementation phase results in a working software product that can be tested for quality and performance.

5. Testing

The Testing phase involves validating and verifying that the software meets the requirements specified in the Analysis phase.

Testers perform various types of testing, including unit tests, integration tests, system tests, and acceptance tests, to identify and fix defects.

This phase ensures that the software is functional, reliable, and ready for deployment. Thorough testing helps to detect and resolve issues early, reducing the risk of defects in the final product.

6. Deployment

In the Deployment phase, the software is released to the production environment and made available to users.

This phase involves installing, configuring, and enabling the software for operational use.

Deployment may be done in stages, such as alpha and beta releases, before the final launch.

The deployment phase also includes user training and documentation to ensure a smooth transition and adoption by end-users.

7. Maintenance

Maintenance is the ongoing phase where the software is updated and improved after deployment.

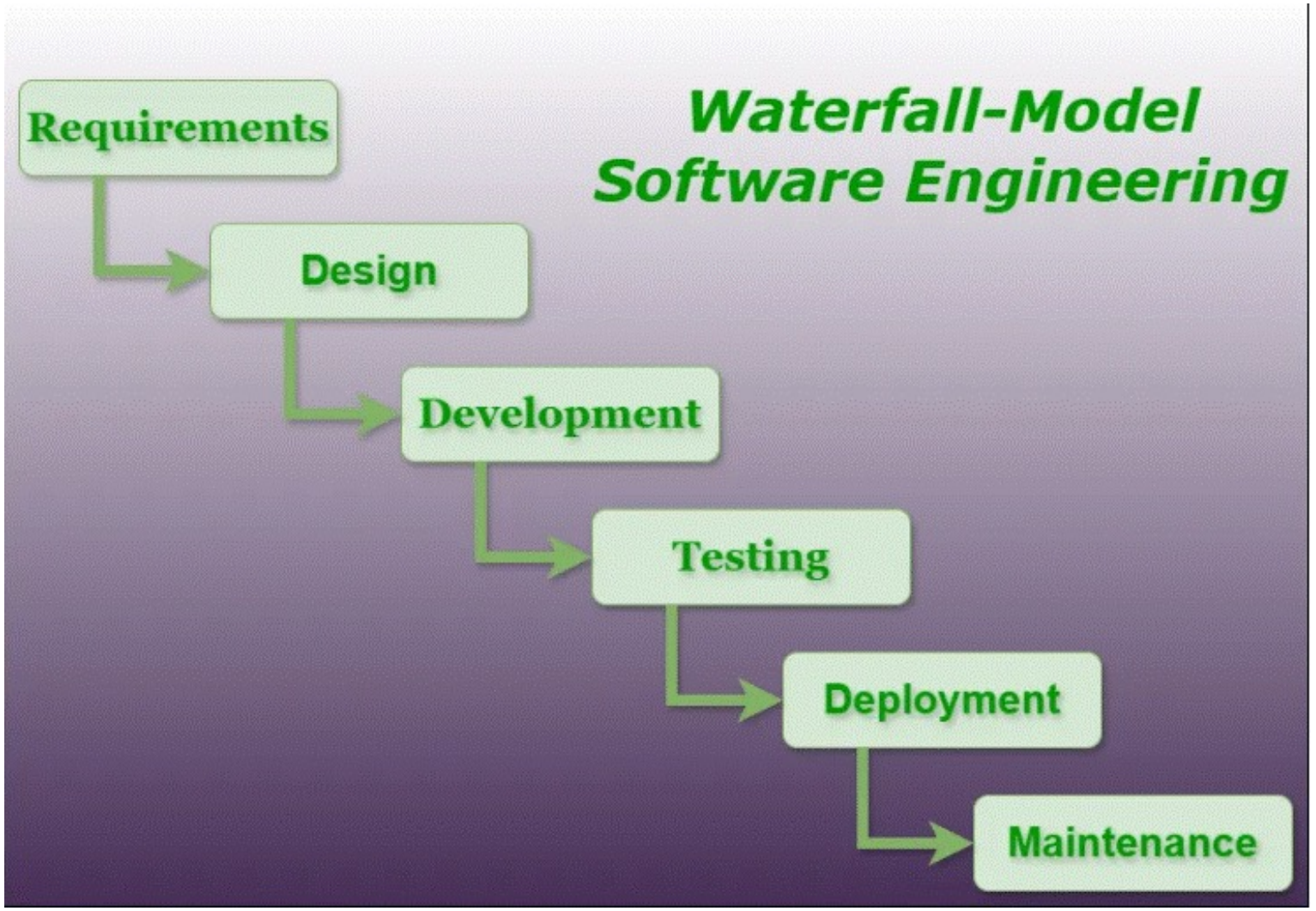
This includes fixing bugs, adding new features, and making performance enhancements based on user feedback and changing requirements.

Maintenance ensures the software continues to meet user needs and operates efficiently over time.

Regular updates and support are crucial to keeping the software relevant and effective in a dynamic environment.

Software Development Methods

- Waterfall
- Rapid Application Development (RAD) and Prototyping
- Agile Methodology
- Continuous Integration/Continuous Deployment (CI/CD)



Waterfall Methodology

Drawbacks

1. Inflexibility to Changes: The Waterfall model is highly sequential and rigid, making it difficult to accommodate changes once a phase has been completed. Any modifications require going back to the initial phases, which can be time-consuming and costly.
2. Late Testing: Testing is only performed at the end of the development cycle, after the entire system has been built. This can lead to the late discovery of critical issues or defects, which are more expensive and challenging to fix.
3. Limited User Involvement: User feedback is typically collected during the initial requirements phase and not incorporated again until the testing phase. This limited user involvement can result in a final product that does not fully meet user needs or expectations, as there is little opportunity for users to influence the design and functionality during the development process.

Rapid Application Development (RAD) and Prototyping

RAD focuses on **iterative** development and the **creation of prototypes** to gather user feedback and refine requirements quickly.

High user involvement is crucial, ensuring that the final product closely aligns with user needs.

This approach reduced development time and increased user satisfaction by incorporating feedback early and often.

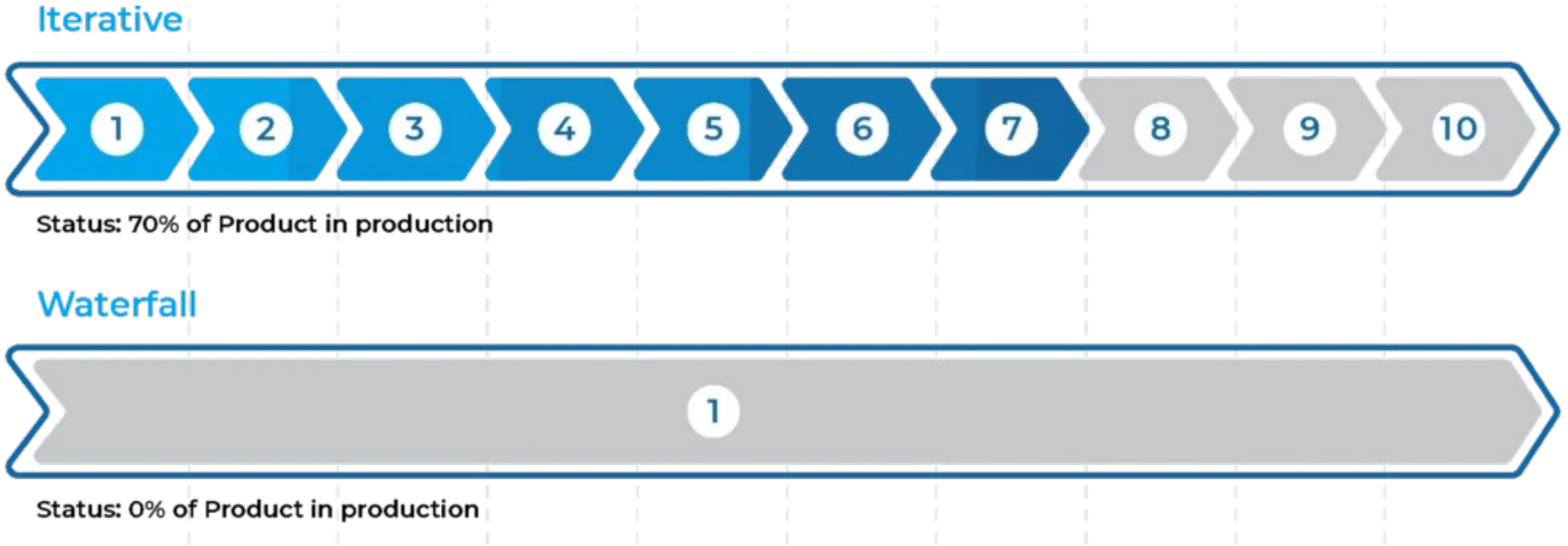
However, the emphasis on speed sometimes led to inadequate documentation and testing, and maintaining active user participation throughout the development process could be challenging.

Agile Methodology: Core Values

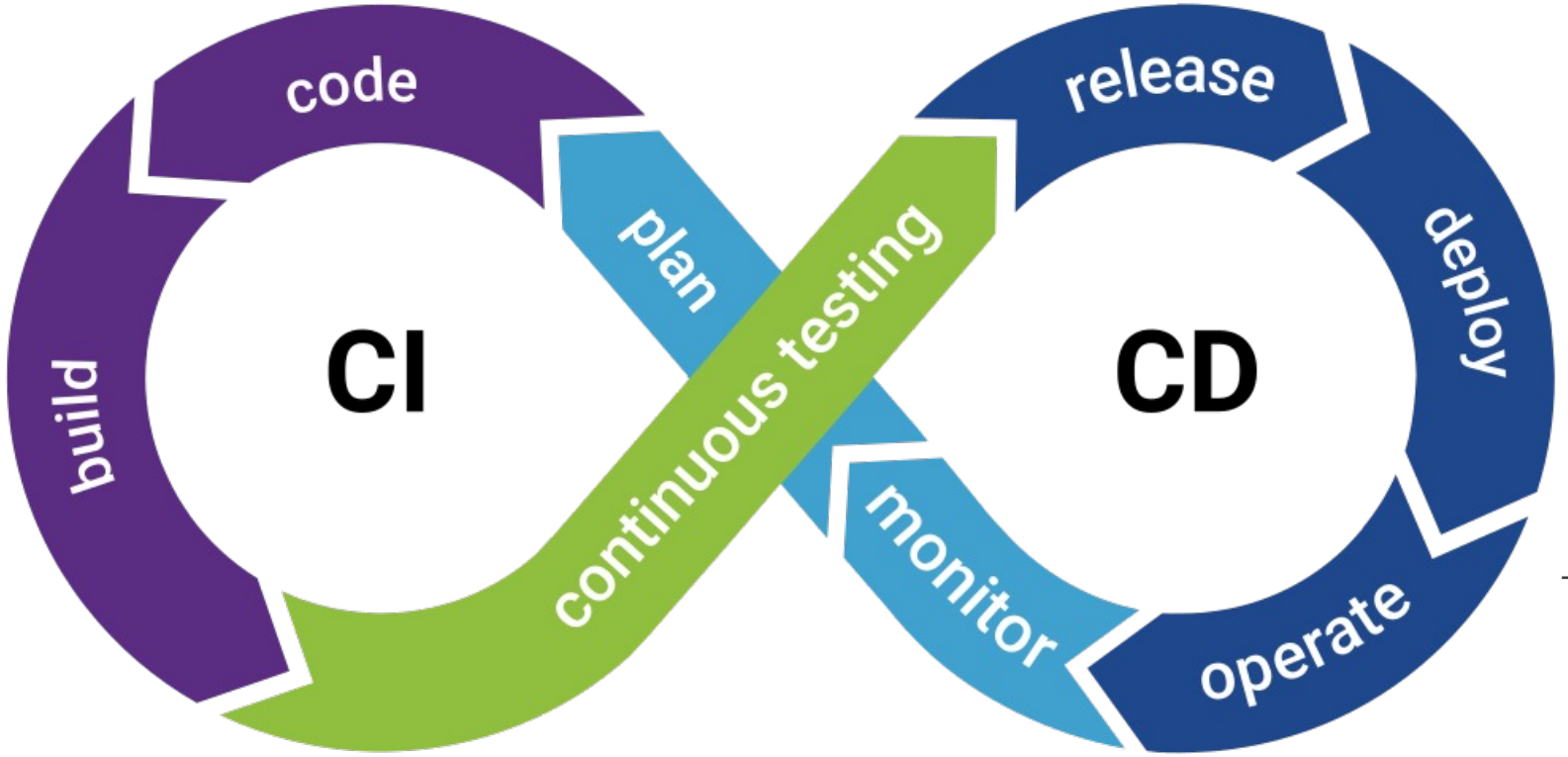
- Individuals and interactions over processes and tools: Emphasizing the importance of effective communication and collaboration among team members.
- Working software over comprehensive documentation: Prioritizing the delivery of functional software over extensive documentation.
- Customer collaboration over contract negotiation: Fostering close and continuous collaboration with customers to ensure their needs are met.
- Responding to change over following a plan: Valuing adaptability and responsiveness to change over adhering strictly to a predefined plan

Iterative Development

- Projects are broken down into small, manageable units of work called sprints or iterations, typically lasting two to four weeks.
- Each iteration involves planning, development, testing, and review, with the goal of producing a potentially shippable product increment.
- This approach allows for continuous feedback and adjustments, ensuring that the project stays aligned with customer needs and market changes.



Iterative vs Waterfall



Continuous Integration/ Continuous Deployment (CI/CD)

Continuous Integration (CI)

- Frequent Code Integration: One of the core principles of CI is that developers should integrate their code changes into the main branch of the shared repository frequently, preferably several times a day
- Automated Builds: Each code integration triggers an automated build process, which compiles the code and packages it into a deployable artifact.
- Automated Testing: CI tools generate detailed build and test reports, providing visibility into the status of the integration process.
- Version Control Integration: CI systems integrate closely with version control systems (VCS), such as Git, Mercurial, or Subversion.
- Continuous Monitoring: Continuous monitoring of the CI process ensures that the system is functioning correctly and that any issues are promptly identified and resolved.

Continuous Deployment (CD)

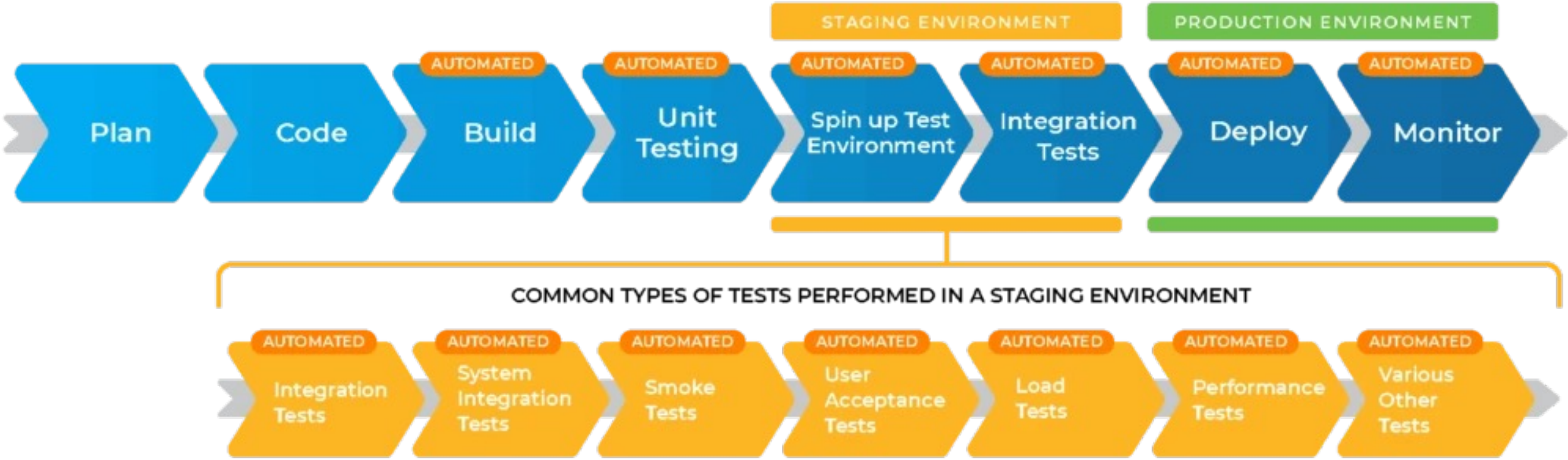
Automated Deployment Pipeline

Incremental Updates and Rollback Mechanism

Monitoring and Logging

Feature Toggles

Automated Software Development



Off-the-shelf Application Packages

- Cost-effectiveness
- Industrial experience and expertise
- Extensively tested and used by numerous other businesses
- Ongoing support and maintenance
- *Lack of customization*
- *Integration issues*

Popular Packages

SAP/Oracle/Microsoft: ERP

Salesforce: CRM

Workday: HR

Microsoft Project

Shopify: E-Commerce

Mailchimp: email marketing

Outsourcing

Outsourcing involves contracting third-party vendors to handle various aspects of the development process, from coding and testing to maintenance and support.

Pros:

✓cost efficiency

✓faster time-to-market

Cons:

❖Collaboration

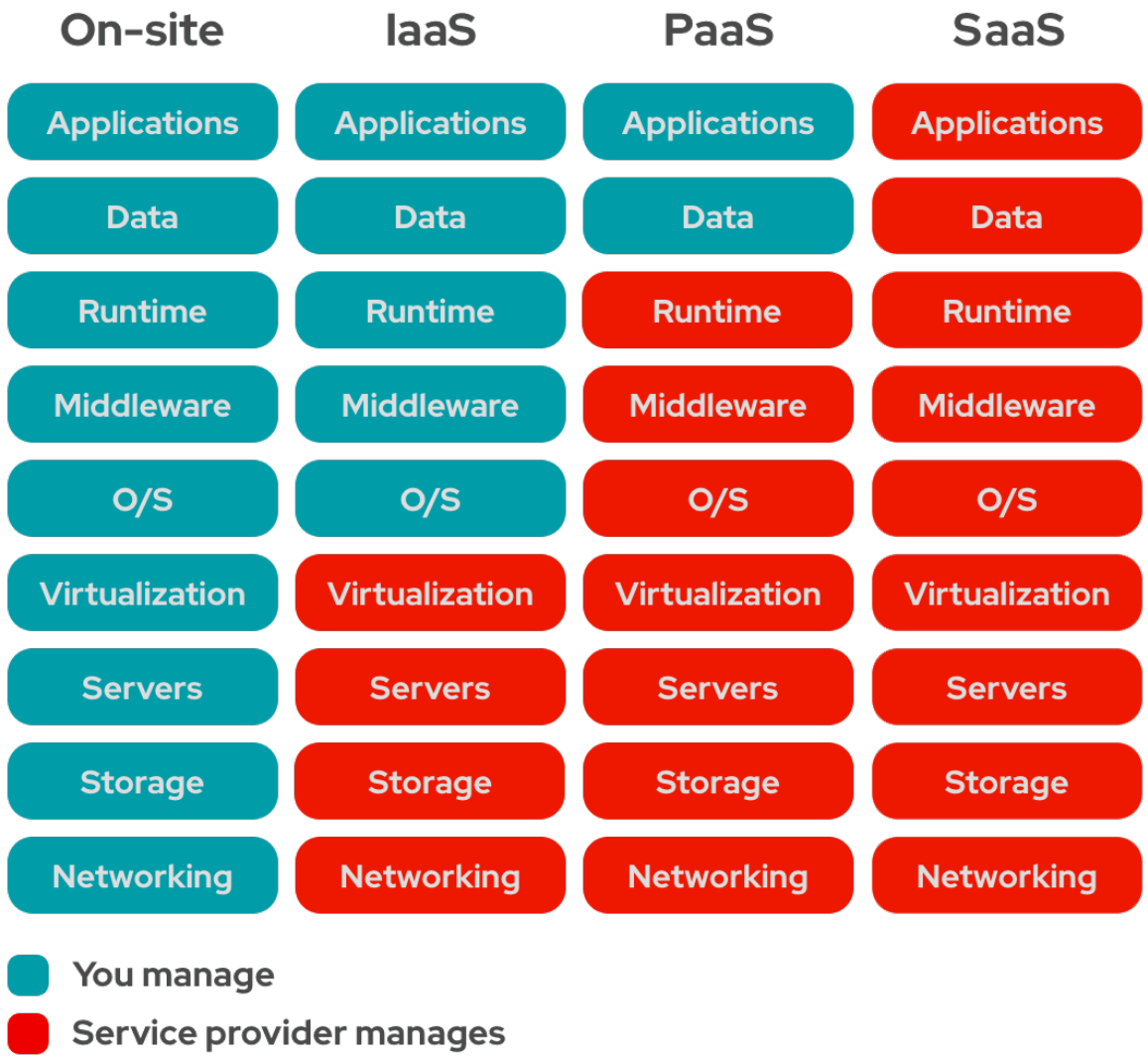
❖Quality control

Cloud Computing

Software as a Service (SaaS) : All functions

Platform as a Service (PaaS): IT services

Infrastructure as a Service (IaaS): basic computing and storage


On-site
IaaS
PaaS
SaaS
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You manage
Service provider manages

In-house or Else?

- A mixed approach for most companies
  - In-house: Core competencies
  - Off-the-shelf or SAAS: General business operation
- Expertise
  - In-house: technology company or not
  - Off-the-shelf or SAAS: for most