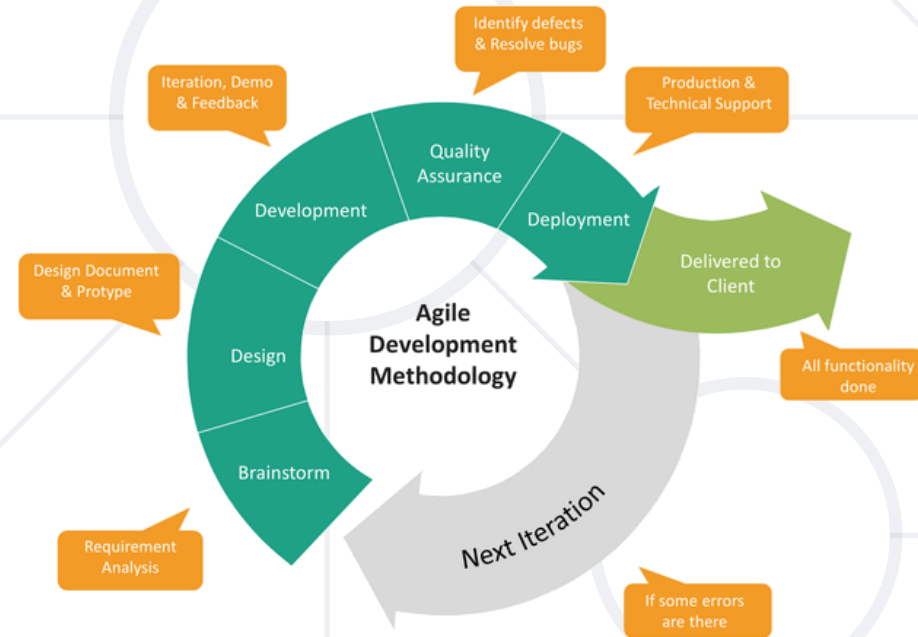


# Software Development Lifecycle and Methodologies



Software  
University  
<https://softuni.bg>



SoftUni



SoftUni Team  
Technical Trainers

# Table of Contents

1. Software Requirements (SRS)
2. Software Development Lifecycle (SDLC)
  - Design, Implementation, Testing, Releasing, Maintenance
3. Software Development Methodologies
4. What is Agile?
5. Scrum
  - Roles
  - Artifacts
  - Events



# Have a Question?

**sli.do**

**#QA-Fund**



# Software Requirements

Understanding the Essentials

- It all starts with **Requirements...**
- **Software requirements** describe the functionality of the software: **what shall it do?**
  - Answer the question "**what?**", not "**how?**"
  - Define **constraints** on the **system**
- Two kinds of **requirements**
  - **Functional** requirements
  - **Non-functional** requirements



- **Requirements analysis** starts from an **idea** about the system
  - Customers usually **don't know what exactly they need!**
  - Requirements come **roughly** → adjusted during the development
  - Requirements **change** constantly!
- Analysis produces some **requirements documentation**
  - **Software Requirements Specification** (SRS)
  - **User stories**
  - **UI prototype**

- The **Software Requirements Specification** (SRS) is a heavy, formal requirements document → rarely used in practice
- SRS describes in detail:
  - **Functional requirements**
    - Business processes
    - Actors and use-cases
  - **Non-functional requirements**
    - E.g. performance, scalability, hardware, integrations, constraints, security, etc.



- It is always **hard** to **describe** and **document the requirements** in a comprehensive way
  - Good requirements **save time and money**, but are hard to reach
- **Requirements always change during the project!**
  - Good requirements **reduce the changes**
  - UI prototypes **significantly reduce changes**
  - Agile methodologies are **flexible to changes**





# More about SRS

## Demo

<https://trello.com/b/5kT2cyJL/requirements>



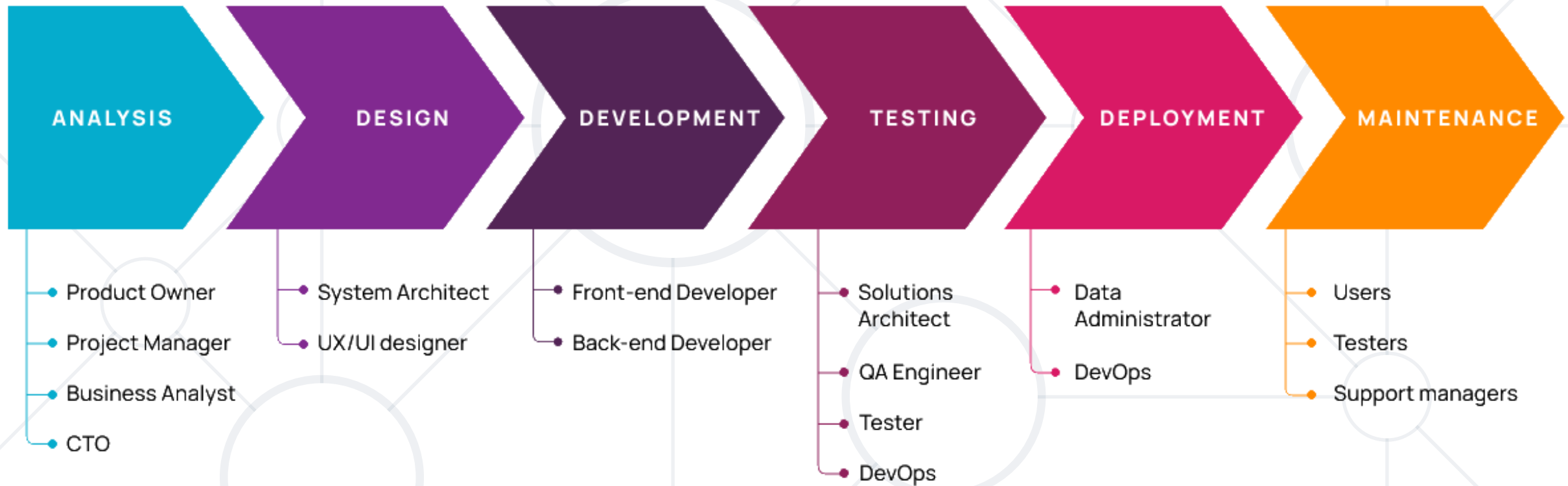
# Software Development Lifecycle (SDLC)

Requirements Analysis, Design, Implementation,  
Testing, Releasing, Maintenance

# Software Development Lifecycle (SDLC)

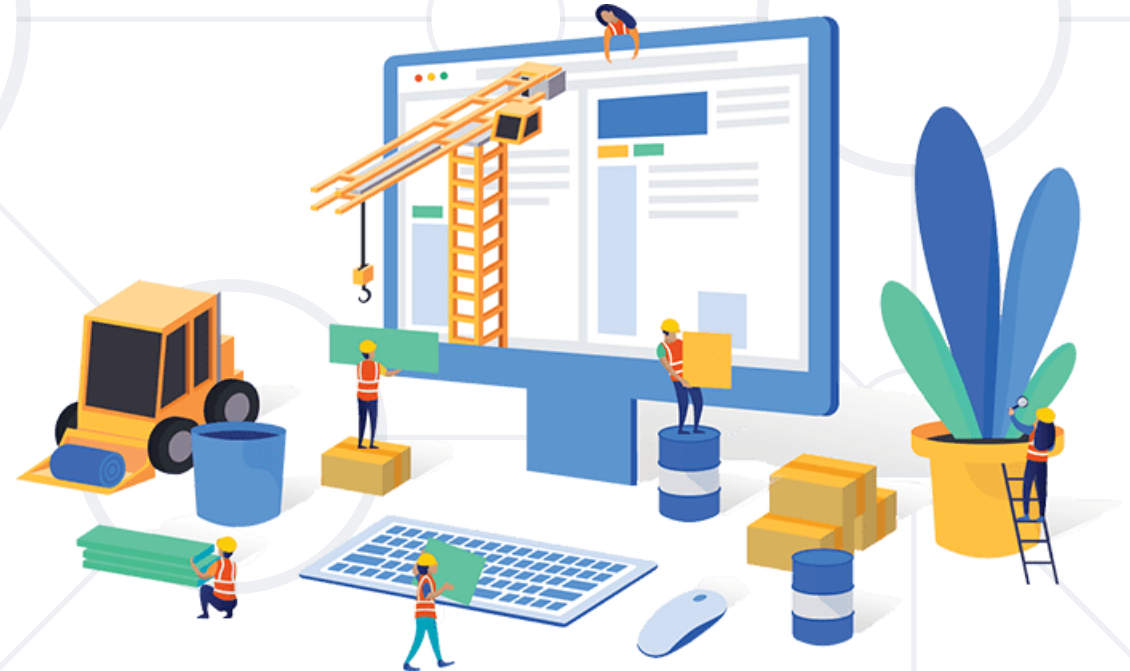
- The **Software Development Lifecycle (SDLC)**
  - **Structured** process
  - **Guides** the development, deployment, and maintenance
  - **Outlines** stages and activities involved in creating software
- **Phases** of SDLC
  - Analyzing and defining software **requirements**
  - Creating software **architecture** and **design**
  - Software **development**
  - Software **testing** and **QA**
  - Software **deployment**
  - Software **maintenance** tasks

# SDLC – Phases and Participants



- **Transition** from **requirement** gathering to **design phase**
  - Essential to address all needs identified in requirement analysis
- **Split** into **System Design** and **Software Design**
- **System Design** focuses on:
  - System **architecture** blueprint creation
  - **Database** design, identifying data resources and relationships
- **Software Design** deals with:
  - Detailed **software specifications**
  - Determining **user interfaces, programming languages**, etc.

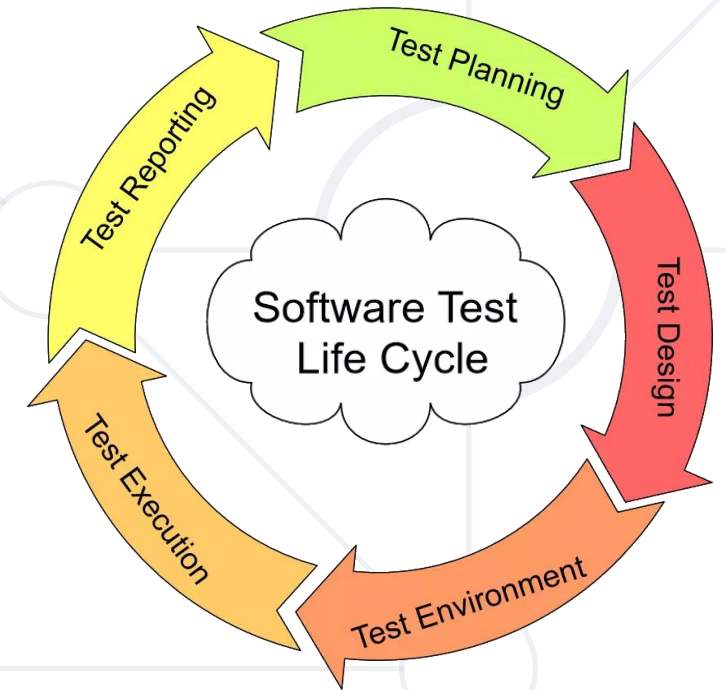
- During the **development phase** developers **build the software**
  - Sometimes it is called "**implementation phase**"
- Software construction **includes**:
  - **Internal method design**
  - **Writing the source code**
  - **Debugging**
  - **Writing the unit tests**
  - **Code reviews** of and inspections
  - **Integration** of classes / modules



- **Coding** is the process of **writing** the programming code (the source code), **running** and **debugging** it
  - The code **follows the architecture and design**
  - Developers perform **internal method design** as part of **coding**
- The **source code** is the output of the **software construction process**
- Concludes with initial testing
  - Developers write unit tests
  - Software prepared for full-scale testing



- **Testing** checks whether the developed software **conforms** to the **requirements**
- Testing aims to **find & report defects** (bugs)
- Software testing process includes:
  - **Test planning**: what, when, how to test?
  - **Test design**: test scenarios & test cases
  - Setup **test environment**: install, configure, prepare test data, ...
  - **Test execution**: perform the tests
  - **Test reporting**: log the test results and bugs found





# Quality Assurance (QA) Engineers

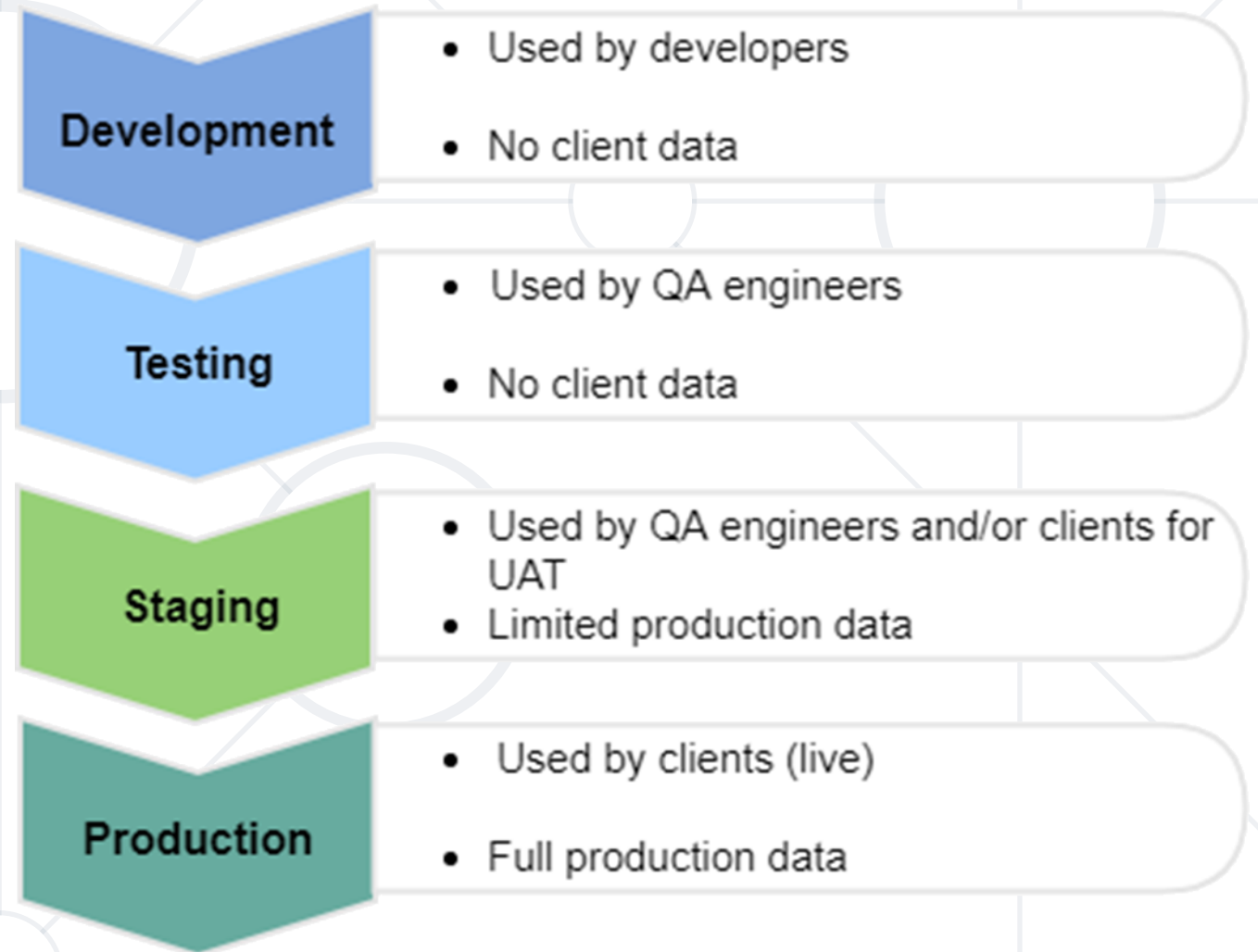
- QA engineers ensure the software quality
- Plan and execute testing activities
  - Test the software, its functionality, UX, etc.
  - Create test plans, design test cases, execute tests
  - Develop and execute test automation scripts
- Report and track bugs and their lifecycle
  - Perform regression testing when bugs are resolved
- Track the development process and its quality
  - Review the requirements, design and code
  - Build and monitor CI/CD pipeline, track QA metrics



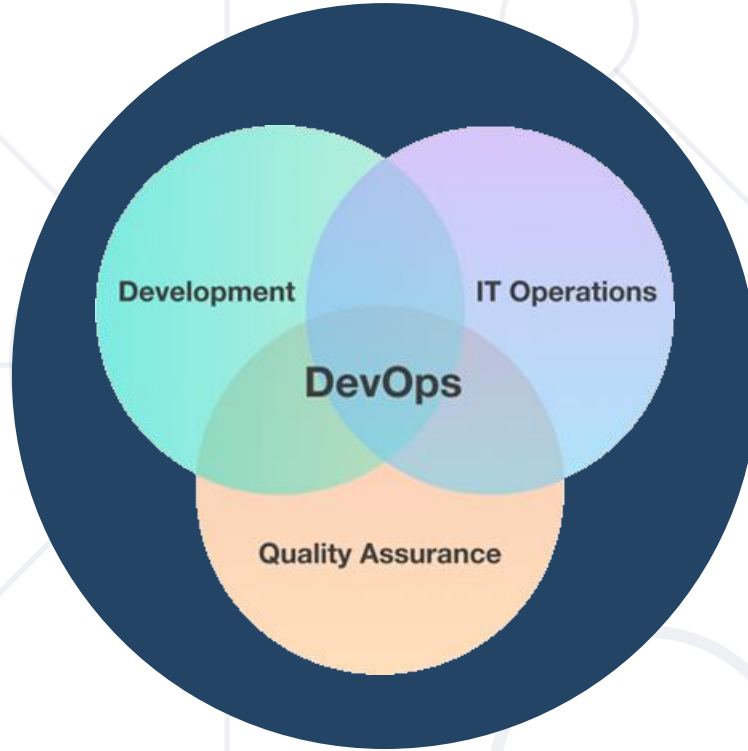
- What is **software deployment**?
  - **Deployment** == getting software out of the **hands of the developers** into the **hands of the users**
  - **Compile, package, install, configure and run** the software into the customer environment / ship the app to the customer
  - **Web app** → deploy the new version to the Web server
  - **Mobile app** → publish a new version in the app store
  - **Desktop app** → release a new version installer to the customers
- **Continuous deployment (CD)**: automatically deploy the software after each commit → ensure the changes are deployable

# Deployment Environments

- Several **environments** for deployment:
  - **Dev environment:** for developers
  - **Test environment:** for QAs
  - **Staging environment:** for QAs and clients
  - **Production environment:** for the end user



- What is **software maintenance**?
  - The process of **changing a system** after it has been released
- **Reasons** for maintenance changes
  - Fixing **bugs** and patching **security vulnerabilities**
  - Changing **business needs**: new features and requirements
  - Adapting to **new environments**: hardware / platforms / software
- Typical **change process**:
  - Analysis → requirements → issue backlog → prioritization
  - For **each fix**: design / re-engineering → code → QA → deploy

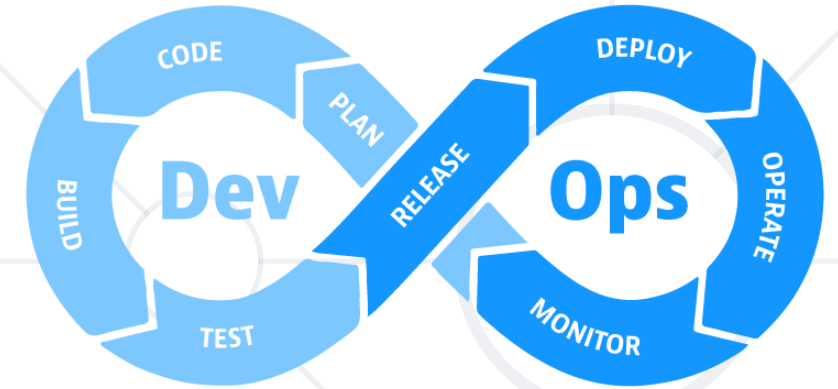


# What is DevOps?

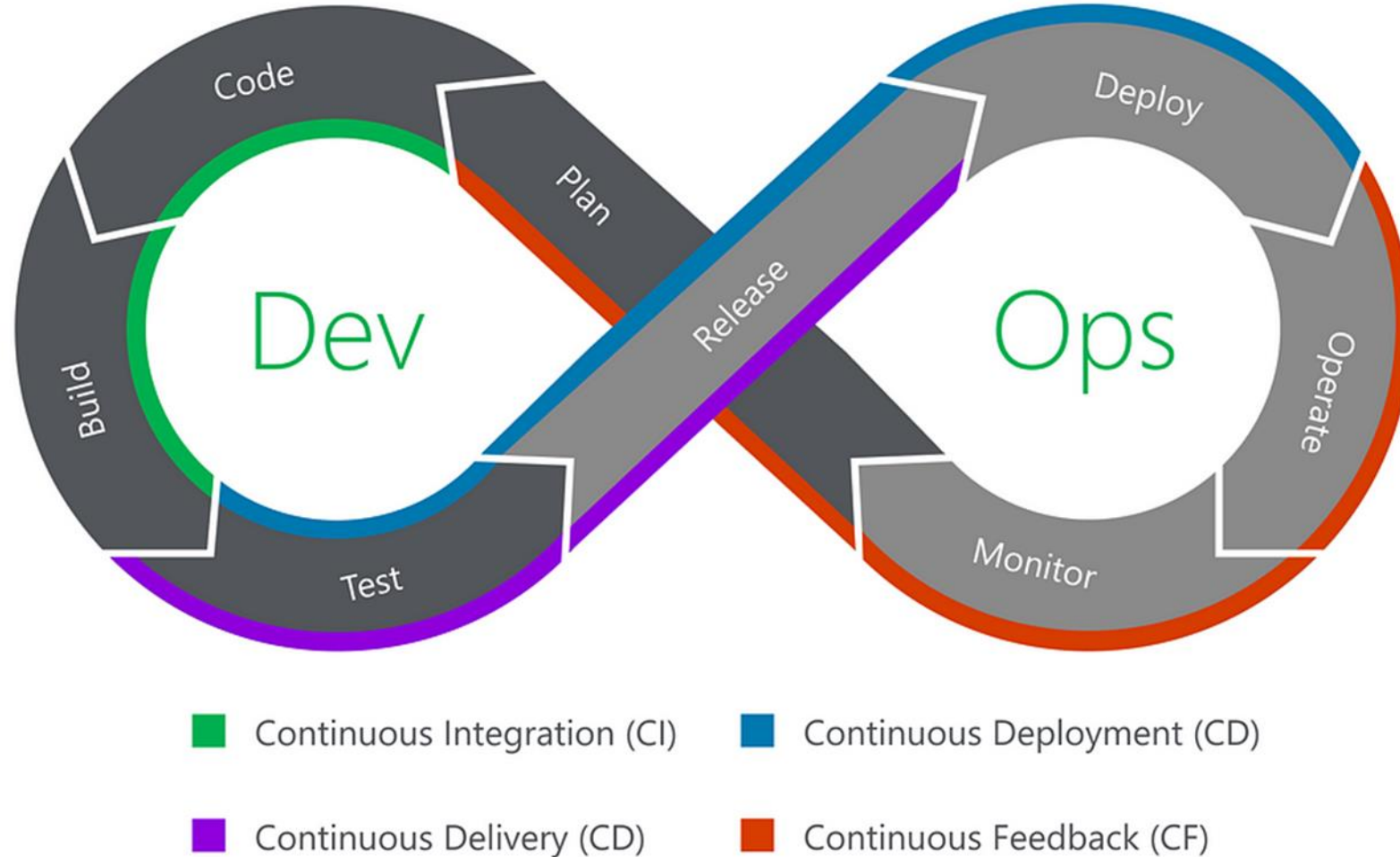
Combining Software Development and IT Teams

# What is DevOps?

- **DevOps** is a set of **practices**, **tools**, and **philosophy** that combines **development** (**Dev**) and **operations** (**Ops**) into one, **continuous process**
- Unites **people**, **process**, and **technology** in application **planning**, **development**, **delivery**, and **operations**
  - Enables **coordination and collaboration** between **isolated roles** like development, IT operations, quality engineering, and security



# Continuous Everything





# DevOps Engineers

- **DevOps engineers** are responsible for the **deployment**, and **maintenance** of software applications
  - **Collaborate** with **development** and **operations** teams
  - Their job includes **automating processes**, **managing infrastructure**, **monitoring performance**, and **ensuring the reliability and security** of the software
- They understand **development lifecycles**, **DevOps culture**, **practices** and **tools**







# Development Methodologies

Heavyweight vs. Agile

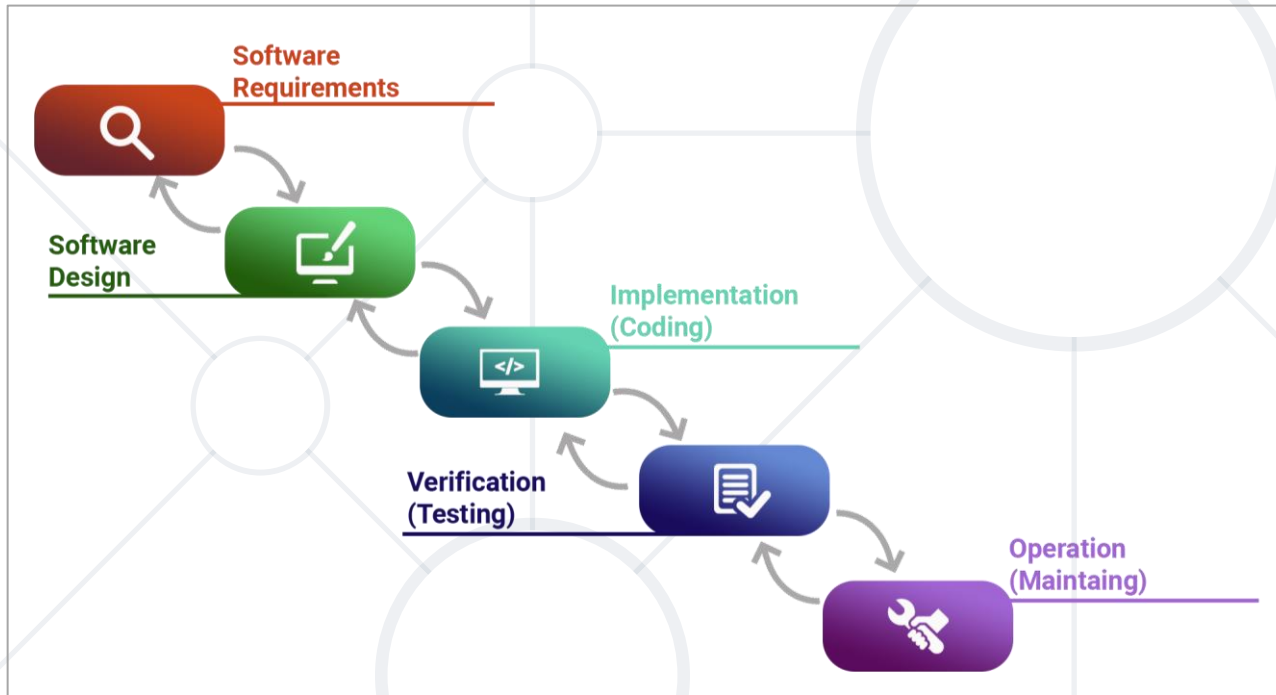
# What is a Development Methodology?

- A development **methodology** is a **set of practices** and procedures for **organizing** the software **development process**
- **Heavyweight (Plan-Driven)** methodologies and **Agile** methodologies
  - Heavy methodologies rely on formal procedures and documents
  - Agile methodologies rely on small iterations and less formalities

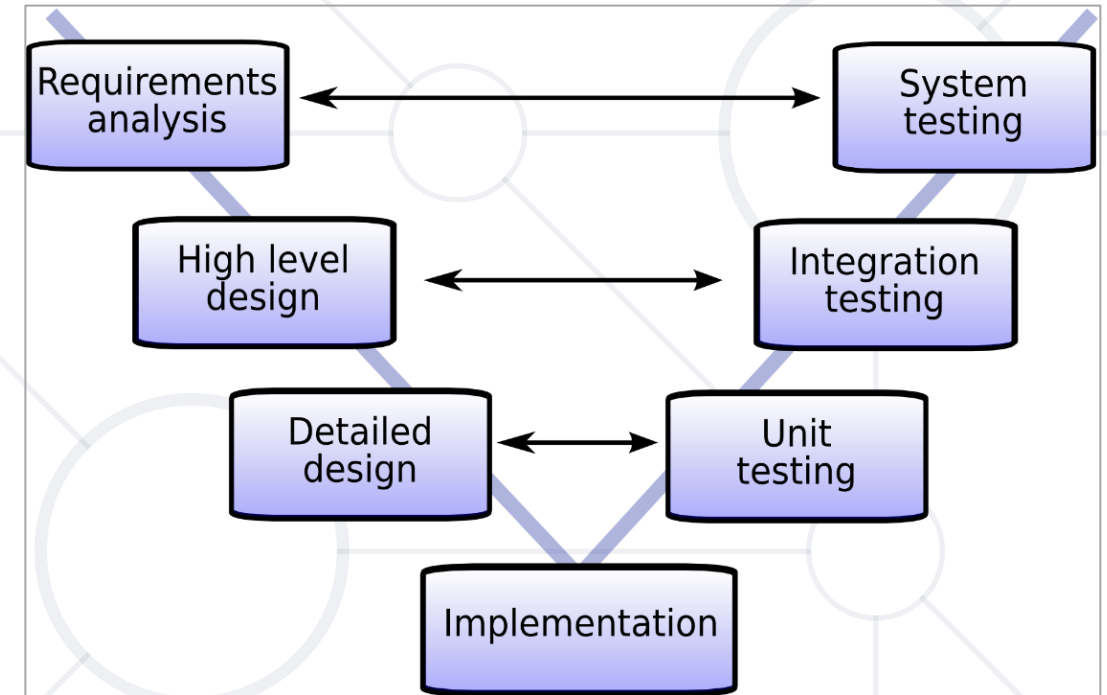


# Heavyweight (Plan-Driven) Methodologies

## ■ Waterfall Model



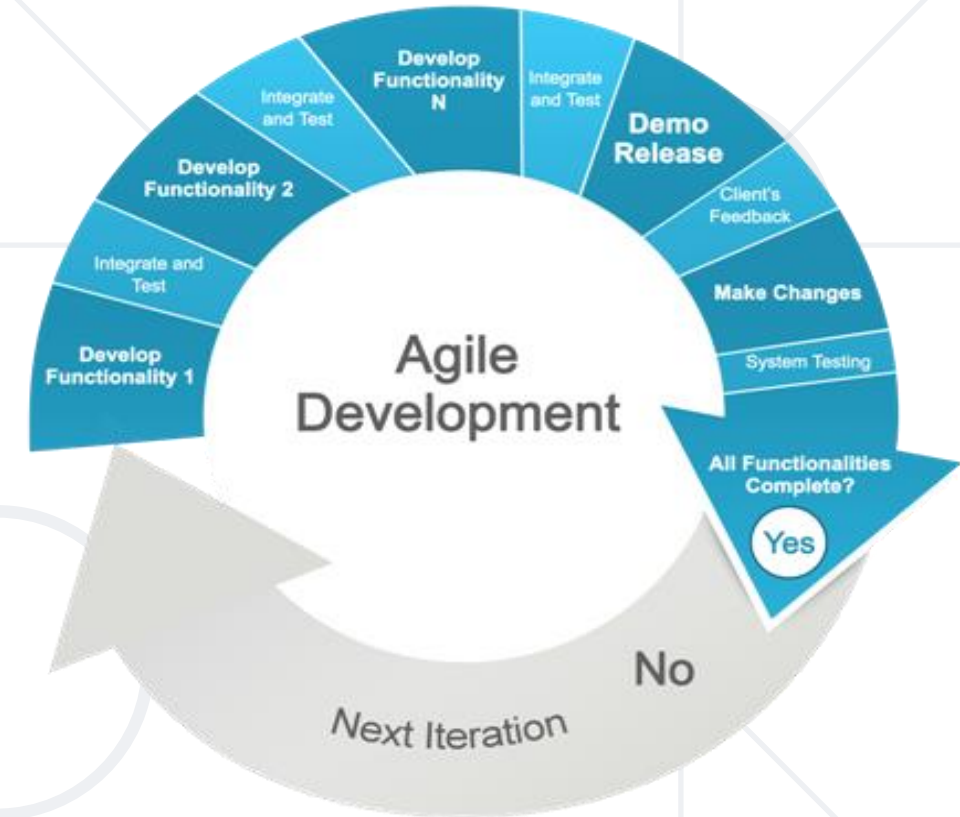
## ■ V Model



## ■ Incremental Model

## ■ Spiral Model

- **Scrum**
- **Kanban**
- Lean Software Development
- Extreme Programming (XP)
- Crystal
- Feature Driven Development (FDD)
- Dynamic Systems Development Method (DSDM)
- Agile Unified Process (AUP)





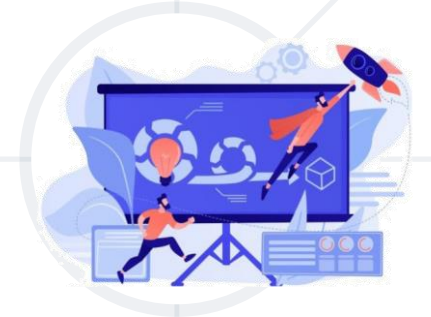
# Agile Development

Lightweight, Incremental, Responding to Change

# The Agile Manifesto and Agile Spirit

- The agile manifesto (Kent Beck, Martin Fowler, Robert Martin, ...)

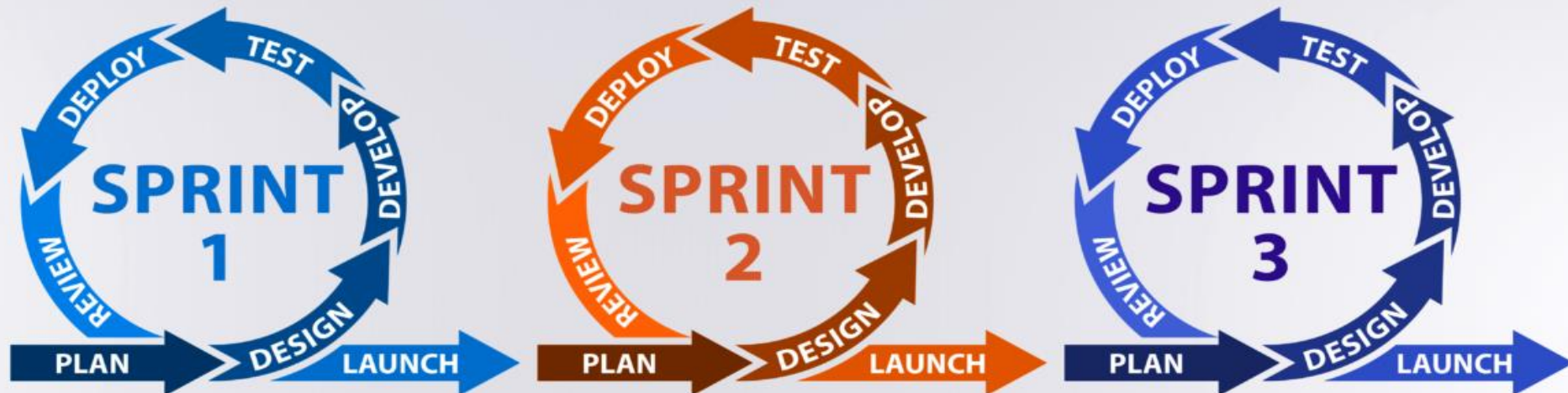
"Our highest priority is to **satisfy the customer** through **early and continuous delivery of valuable software**"



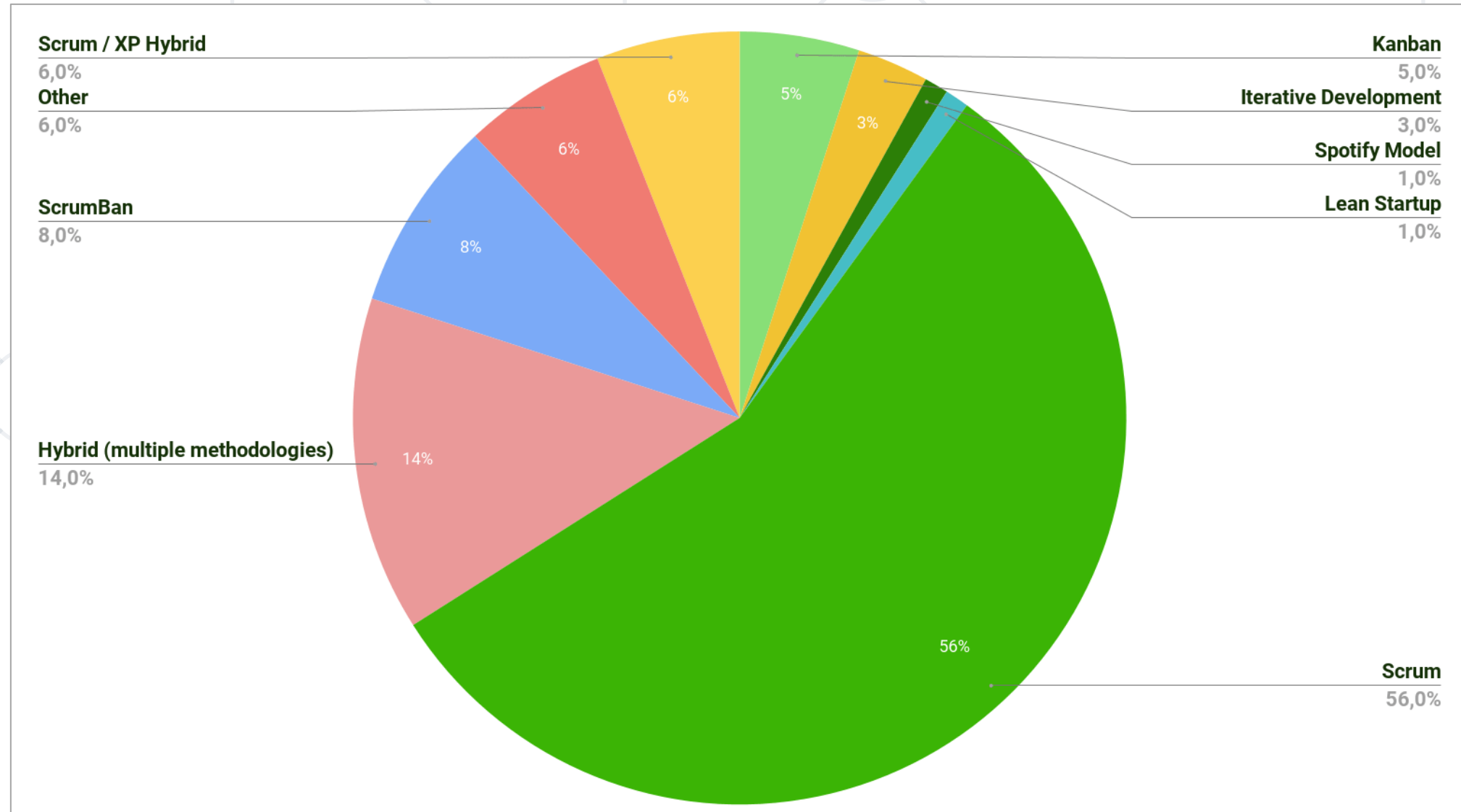
- The **agile** spirit:
  - Incremental: **working software frequently** over heavy documentation
  - Cooperation: **customer collaboration** over contract negotiation
  - Straightforward: **individuals + interactions** over processes + tools
  - Adaptive: **responding to change** over following a plan

# Iterative and Incremental Development

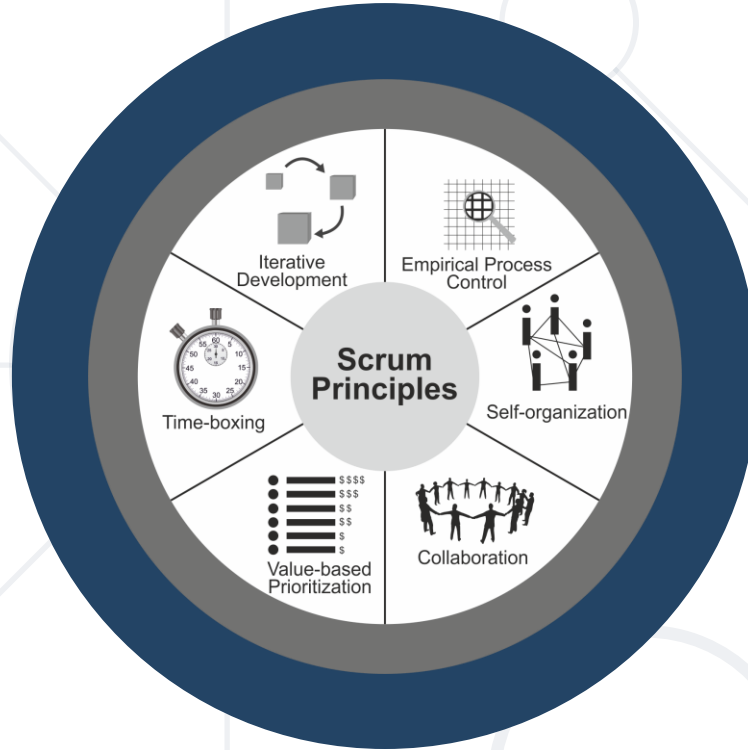
- Agile development relies on **increments** (steps)
  - **Small portions** of software, developed in iterations
- Each **iteration** goes through its own dev **lifecycle**:
  - requirements → design → code → test → deploy → review → repeat



# Most Popular Agile Approaches







# Scrum

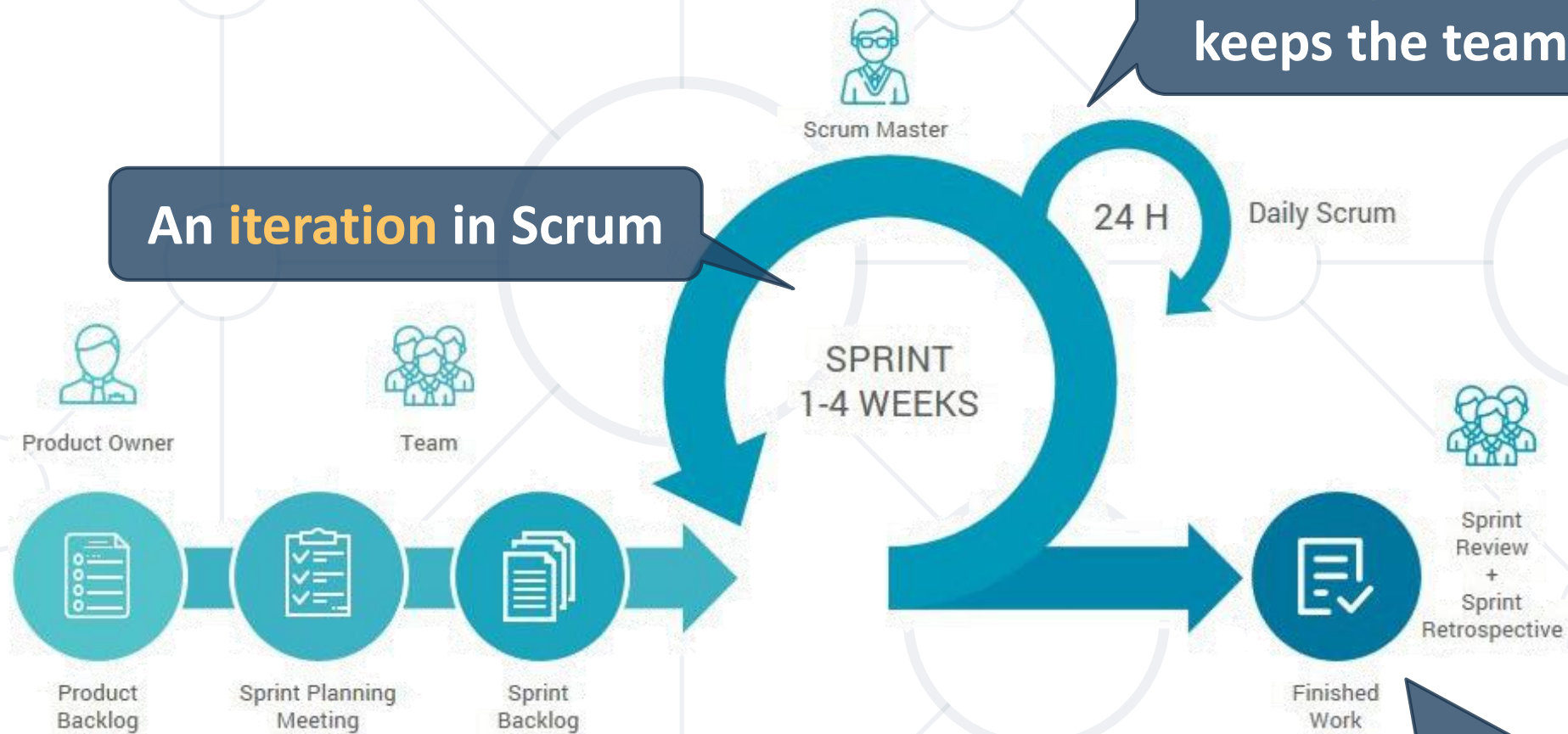
## What is SCRUM?

- **Scrum** is an iterative and incremental **agile software development methodology** for product development
- Based on **short cycles** (2-4 weeks)
- Iterative approach
- **4 basic steps** while running each cycle:
  - Step 1: **Planning & Estimation**: Sprint planning and creating a backlog
  - Step 2: **Implementation**: Running the sprint
  - Step 3: **Review**: Product review and Demo
  - Step 4: **Retrospective**: Next sprint planning
  - ...and then repeat again

# Scrum Practices

The **daily Scrum meeting** keeps the team in sync

An **iteration** in Scrum



All **features** that have to be **developed**

All **features** planned for the **current sprint**

Sprints end with **working features + review meeting**



# Scrum Roles

## Who?

# Scrum Team - Roles

- Scrum handles **three basic roles**:

**Build the right thing**

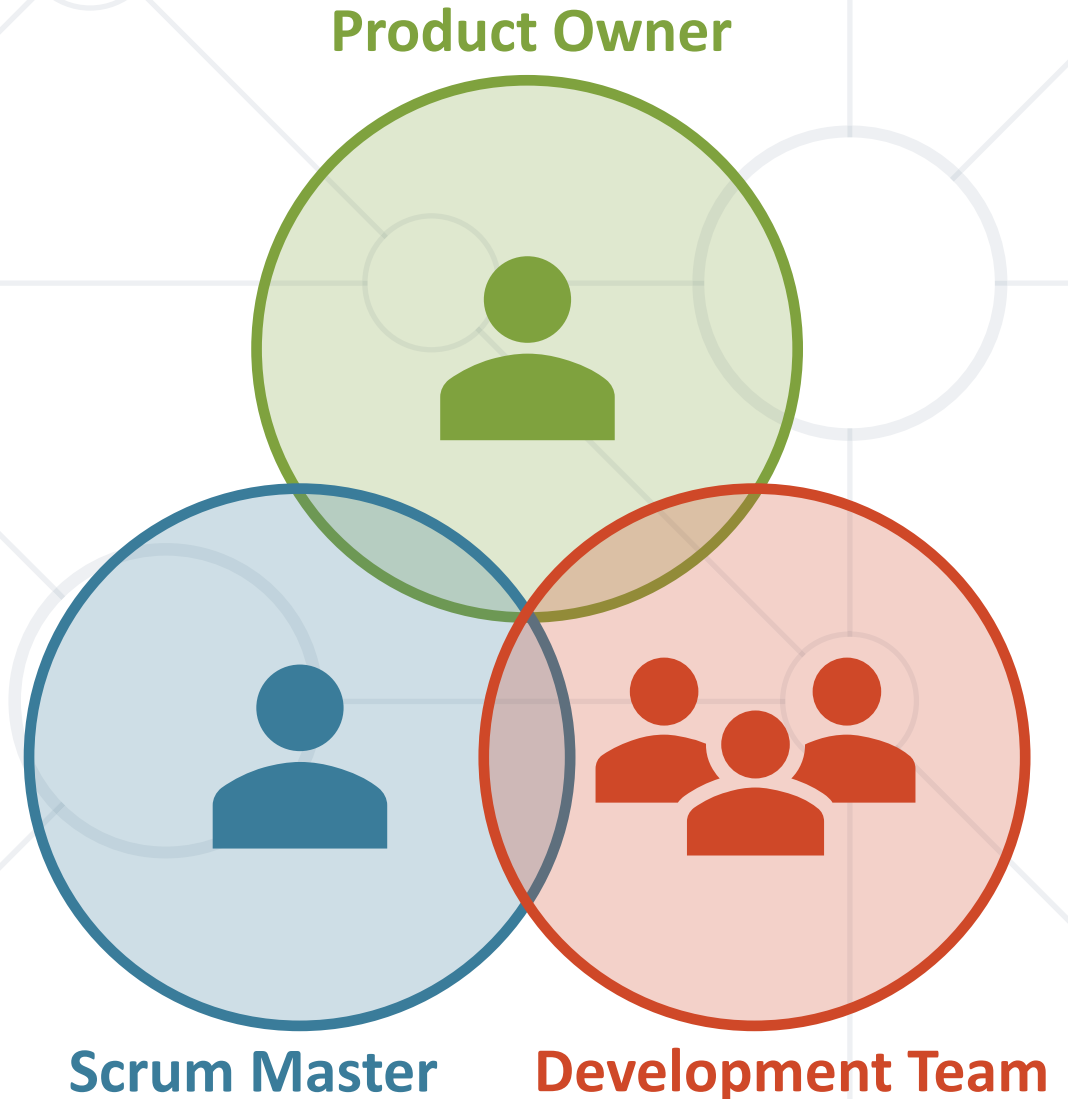
Manages the Product Backlog  
Optimizes value of the Product

**Build the thing right**

Manages the Scrum process  
Removes Impediments

**Build it fast**

Manages itself  
Creates "Done" Increments



# The Product Owner

- Provides **product vision** and **aligns development** with **business value**
- Solely responsible for **Product Backlog management**
- **Prioritizes** and refines **Product Backlog**
- **Facilitates communication** between the team and stakeholders
- **Represents customer interests** and maintains customer-centric approach
- **Leads product strategy** based on market trends and competitive analysis
- Serves as **single point of accountability** for product decisions



# The Scrum Master

- Serves as **facilitator** and **coach** for the Scrum team
- **Ensures Scrum principles** and **practices** are adhered to
- **Removes obstacles** that hinder team progress
- **Facilitates** Scrum **events** and **promotes effective communication**
- **Fosters** a collaborative and **self-organizing team environment**
- **Acts as** a **buffer** between the team and external interference
- **Leads** the organization's Scrum adoption and provides Scrum training
- **Promotes** understanding of Scrum theory, values, rules, and practices
- Works to **enhance** the **productivity** of the team by optimizing interactions and processes

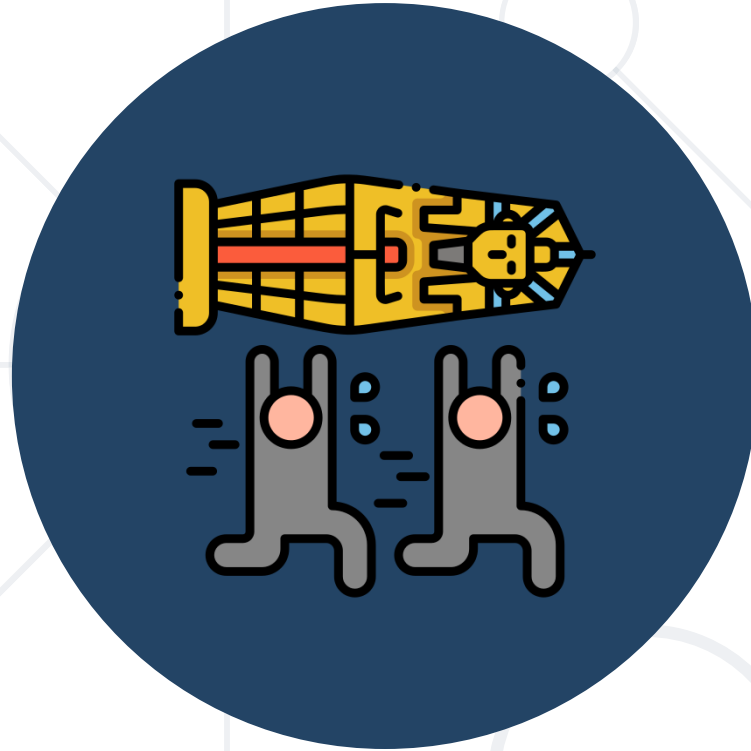


# The Development Team

- Committed to creating a usable product Increment each Sprint
- Responsible for **creating** the **Sprint Backlog** and plan
- Upholds quality by **following** a **Definition of Done**
- **Adapts plan daily** to achieve the Sprint Goal
- **Composed** of **3-9 individuals**, without sub-teams or specific titles
- **Cross-functional** and **self-organized**, having all skills necessary within the team
- Individual skills present, but the **focus** is on **team accountability**
- Encourages **knowledge sharing** and **collaborative problem solving**



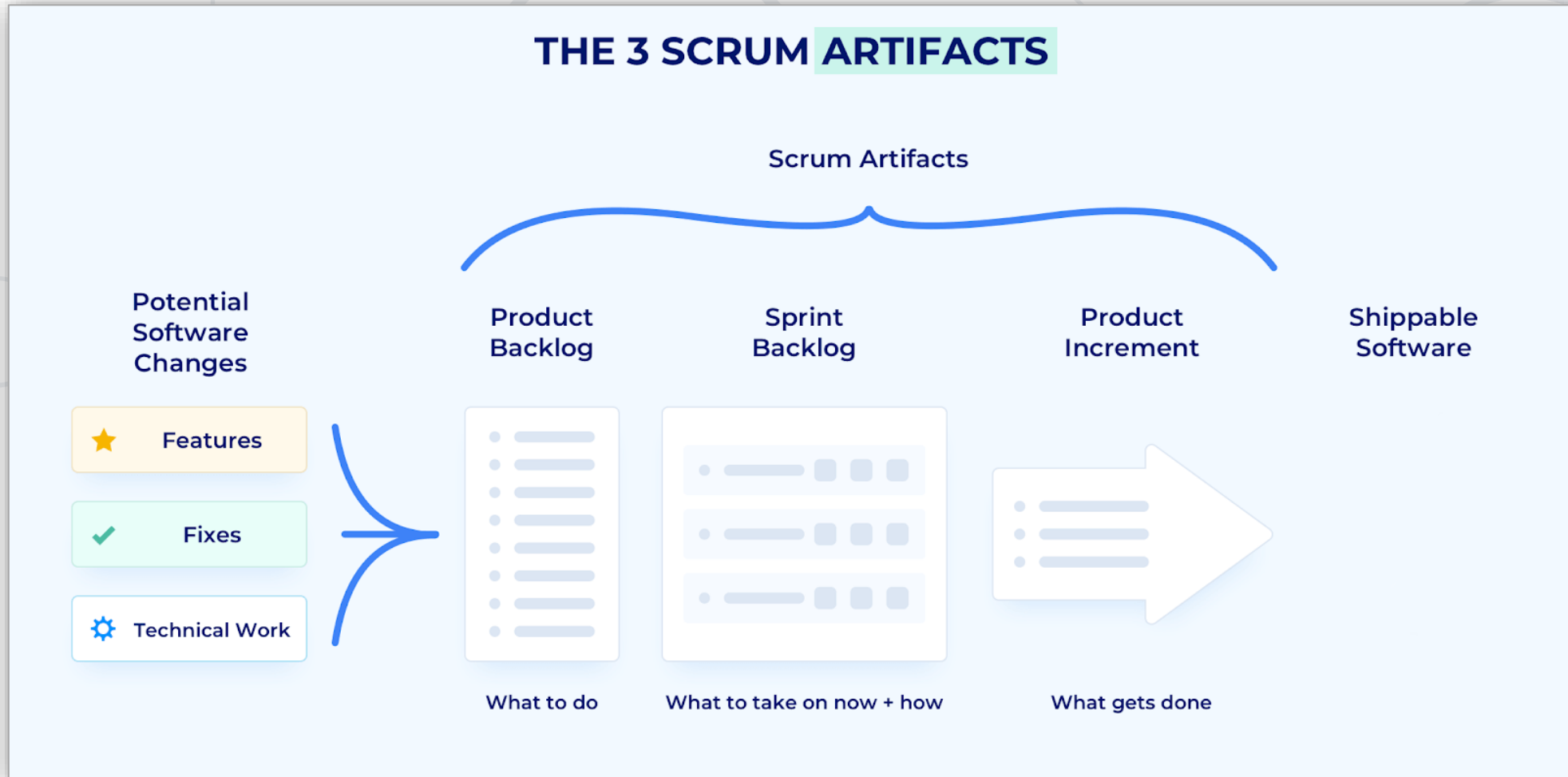




# Scrum Artifacts

## How?

- Scrum uses **three artifacts to manage work**:



# The Product Backlog

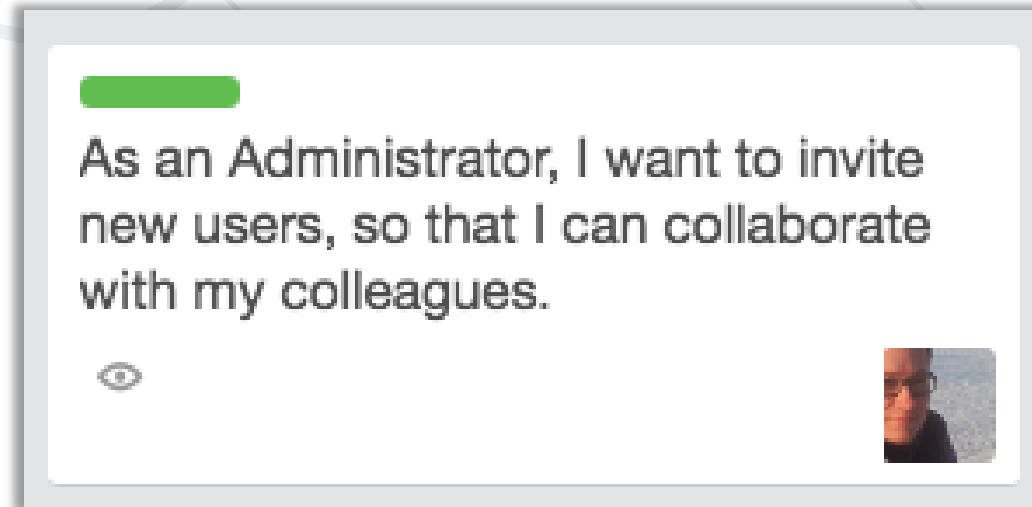
- List of **all** desired product **features, changes** to features, **technical improvements**, or anything **valuable** to the product
- **User stories** are commonly used items, outlining the needs and values of the user
- **Acceptance criteria** are attached to user stories to clearly define when they are considered 'done'
- A **living document**, always reflecting the most accurate and important tasks for the team
- Items are ordered based on their **value, risk, priority**, and **necessity**
- **Each item** should be **clear to everyone** on the team




- **Formal specifications** are too heavy
  - Do not work well in **dynamic projects** where business requirements change every day (most projects)
- Agile development needs **agile requirements**
  - Pieces of functionality, implemented in **many small iterations**
- How agile requirements are **structured**?
  - Using **simple**, informal descriptions of **features (stories)**
  - **User story**: a small feature that brings value to the end-user

# What is User Story?

- **User story** (describes a software feature)
  - A **user** needs to **accomplish something**
  - Written **informally** (in words / images / sketches)
- **User stories** have:
  - **Actor** (who?)
  - **Goal** (what?, why?)
  - **Other details**
    - Owner, estimate, constraints, diagrams, attachments, ...




# User Story – Structure and Example



**As a** <role>  
**I want to** <goal>  
**so that** <benefit>

**Acceptance criteria:**  
(conditions of satisfaction)

...  
...



**As an** *account manager*  
**I want** *a sales report of my accounts*  
*to be sent to my inbox daily*  
**so that** *I can monitor the sales*  
*progress of my customer portfolio*

**Acceptance criteria:**

1. The sales report is sent daily to my inbox.
2. The report contains the following details: ...
3. The report is in MS Excel format.

### Writing a user story

- 1 Define your **end user**
- 2 Specify what **they want**
- 3 Describe **the benefit**
- 4 Add **acceptance criteria**

# More about User Stories

## Demo

<https://trello.com/b/6lwaEu3F/web-store-project>

- Set of criteria which a User Story (US) should meet in order to be completed
- There is no set rule of who is defining the Acceptance Criteria
- In general, the Acceptance Criteria are initiated by the Product Owner, but a result of the work of the entire (Scrum) team
- A good practice is that each US has at least one Acceptance Criteria (AC)
- There is no maximum cap of AC-s





# Acceptance Criteria (2)

- Another perspective – you can tell a better story when using clear set of AC-s.
- Help's avoid misconception, misleading and confusion.
  - An example of Acceptance Criteria:

**User story:** *As a user, I want to be able to register online, so that I can start shopping online.*

**Acceptance criteria:**

- ☐ User can only submit a form by filling in all required fields
- ☐ The email user provided must not be a free email
- ☐ Submission from same IP can only be made three times within 30 minutes
- ☐ User can only submit a form by filling in all required fields
- ☐ User will receive a notification email after successfully registration

# The Product Backlog - example

PRODUCT BACKLOG EXAMPLE						
	As a...	I want to be able to...	So that...	Priority	Sprint	Status
1	Administrator	see a list of all members and visitors	I can monitor site visits	Must	1	Done
2	Administrator	add new categories	I can allow members to create engaging content	Must	1	Done
3	Administrator	add new security groups	security levels are appropriate	Must	1	Done
4	Administrator	add new keywords	content is easy to group and search for	Must	1	Done
5	Administrator	delete comments	offensive content is removed	Must	1	Done
6	Administrator	block entries	competitors and offenders cannot submit content	Must	1	Done
7	Administrator	change site branding	the site is future-proofed in case brand changes	Could	1	Done
8	Member	change my password	I can keep secure	Must	1	Done
9	Member	update my contact details	I can be contacted by Administrators	Must	2	Work in Progress
10	Member	update my email preferences	I'm not bombarded with junk email	Should	2	Work in Progress
11	Member	share content to social networks	I can promote what I find interesting	Could	2	Work in Progress
12	Visitor	create an account	I can benefit from member discounts	Must		To be started
13	Visitor	login	I can post new entries	Must		To be started
14	Visitor	add comments	I can have a say	Must		To be started
15	Visitor	suggest improvements	I can contribute to the site usability	Should		To be started
16	Visitor	contact the Administrators	I can directly submit a query	Could		To be started
17	Visitor	follow a member's updates	I'm informed of updates from members I find interesting	Should		To be started
18	Visitor	view a member's profile	I can know more about a member	Must		To be started
19	Administrator	generate incoming traffic report	I can understand where traffic is coming from	Must		To be started

# The Sprint Backlog

- It **covers** only the **current Sprint**
- It reflects the US-s which are selected for the Sprint only
- It is sole **managed** by the **Development Team**
- While the Product Backlog lists ALL User Stories required for the project, the Sprint Backlog would reflect only the **User Stories** for the **current Sprint**
- The **Priorities**, the **Estimates** and the number of **User Stories** to be included in the Sprint (in the Sprint Backlog) should be rightfully reflected



# Product Backlog vs. Sprint Backlog

## 'Product Backlog' vs 'Sprint Backlog'

*The key differences between two common artifacts.*



Anything that needed to accomplish the project vision



Product Owner owns the Product backlog



Contains requirements, defects, tasks.



Everyone contributes to the product backlog



Product backlog refinement meeting is to refine the product backlog

Anything that is needed to fulfil the sprint goal.



Development team owns the sprint backlog



Subset of product backlog items defined as priority by product owner.



Only development team contributes to the sprint backlog



Sprint Planning meeting is to refine the sprint backlog items



# Product Increment

- Also referred as "**potentially releasable product**".
- Why is considered as "product increment"?
  - The key to remember – the "product increment" must function on its own
  - It is not the final version (alpha, beta, delta....)
  - The next version is an improvement of the previous one
  - Almost a never-ending cycle



- The Definition of Done (DoD) is based on the agreed criteria that must be met for a team to consider an aspect of the product shippable or complete.
- It established a shared understanding across the team what must be done for a user story to be considered finished.

## DONE

adjective

1. Unit tests passed
2. Code reviewed
3. Acceptance criteria met
4. Functional Tests passed
5. Non-Functional requirements met
6. Product Owner accepts the User Story

# Definition of Done – example

**DEFINITION OF DONE**

Team: \_\_\_\_\_  
Product: \_\_\_\_\_

As a team, before saying that an item of the sprint backlog is Done, we agree that it will meet the following:

- ☒ Code is complete and according to development team standards.
- ☒ Code refactored.
- ☒ Meet acceptance criteria.
- ☐ Code checked-in to the repository.
- ☒ Unit test written and green.
- ☐ Test coverage: \_\_ %.
- ☐ Pair programming.
- ☒ Peer review.
- ☐ Code merge and tagged.
- ☒ Deployed to the development environment.



## **Scrum Events**

Planning, Executing, Reviewing, and Improving



# Sprint planning

- It all starts with planning
- At the beginning of each iteration in the form of a meeting
- The **input** for the Spring Planning is the Product Backlog
- The whole "Scrum Team" participate
- It should not take more than 2-4 hours
- It should cover ALL for the upcoming sprint
- The **output** is the Sprint Backlog



# Daily Scrum

- The most important event during the delivery phase
- Often referred to as "Daily Stand Up"
- ...and yet, it should not take more than 15 min
- Answers the three main questions:
  - What did I do yesterday?
  - What am I planning to do today?
  - Are there obstacles ahead of me?
- Daily Scrum Explained
- An event for the Dev Team!



# Review

- Time: Held at the end of each Sprint
- Duration: **4 hours** for 4-week Sprints, proportionally **shorter** for shorter Sprints
- Who: **Scrum Team, Customers, Key stakeholders**
- The time that the team shows what has been produced during the sprint
- It is also meant to inspect the Increment and adapt the Product Backlog if necessary
- During the meeting, the team collaborates on what steps should be taken next to optimize the value of the software



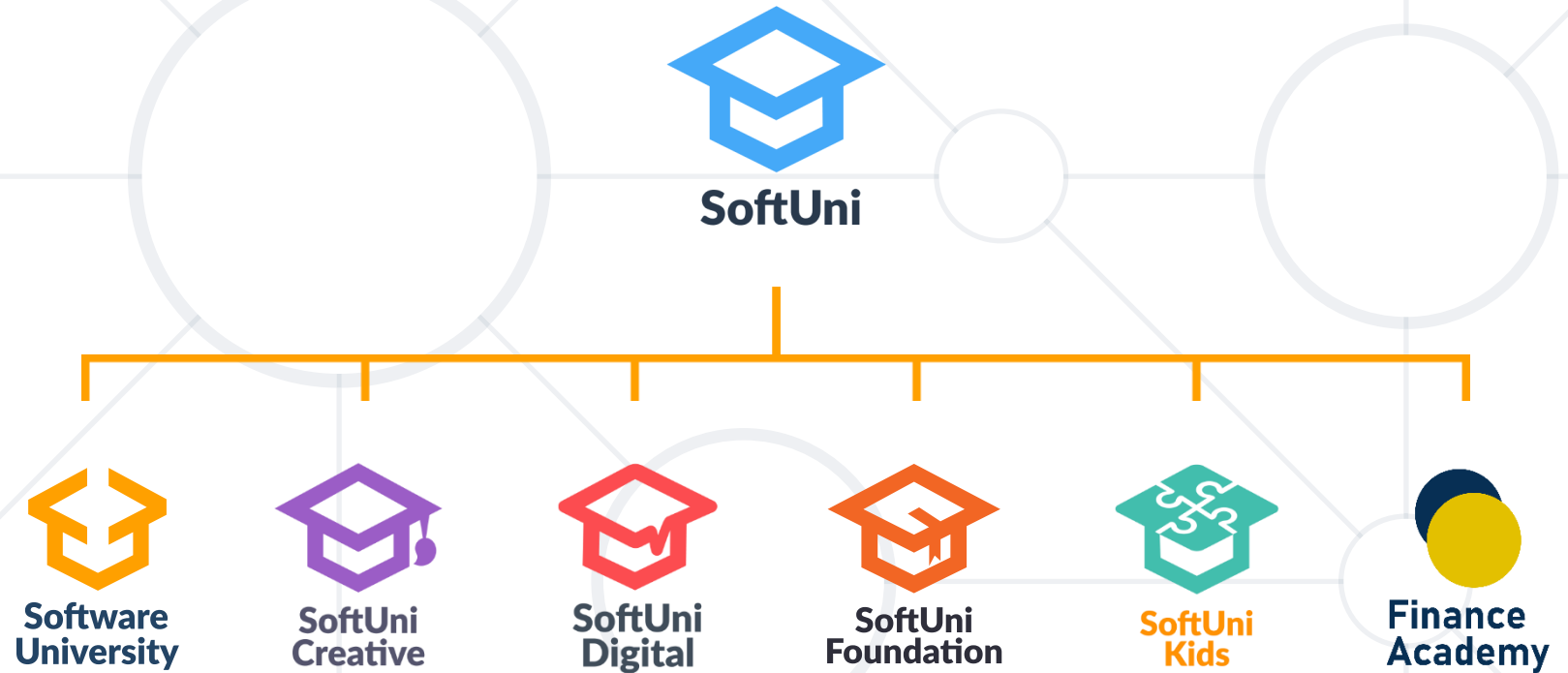
# Retrospective

- When: Immediately following the Sprint Review
- Duration: up to **3 hours** for 4-week Sprints, proportionally **shorter** for shorter Sprints
- Goal: Inspect and adapt with regard to **people, relationships, processes, and tools**
- The Scrum Team reflects on the work done during the sprint
- Typical Questions to be asked:
  - What did go well?
  - What did not go well / What should be changed?
  - Unclear Areas?
  - New Suggestions / New ideas?



- Everything starts with **requirements**?
- **SDLC** phases:
  - **Requirements, Design, Implementation, Testing, Releasing, Maintenance**
- What is **DevOps**?
- Software Development Methodologies
  - **Agile**
  - **Scrum**
    - **Roles** – Product Owner, Scrum Master, Development Team
    - **Artifacts** – Product Backlog, Sprint Backlog, Product Increment
    - **Events** - Planning, Daily Scrum, Review, Retrospective

# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**



**POKERSTARS**  
POKER | CASINO | SPORTS  
a Flutter International brand

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

  
**DRAFT  
KINGS**



**SOFTWARE  
GROUP**

createX



**Postbank**  
Решения за твоето утре



**BOSCH**

**DXC**  
TECHNOLOGY



**SmartIT**





- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



Software University

