SDLC and Agile

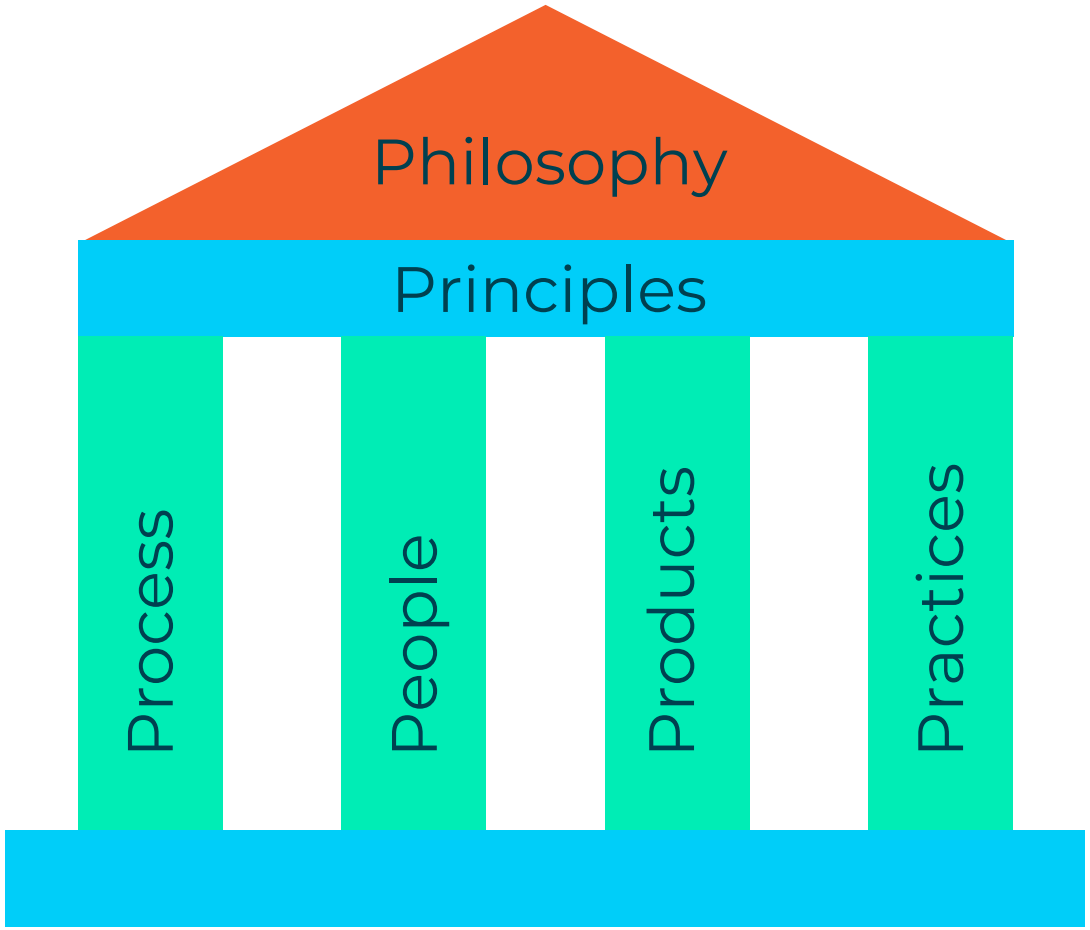
What is Systems Development?

> *"The process of taking a set of business requirements through a series of structured stages and translating them into an operational IT system."*
>
> 'Developing Information Systems'

- **Can we do without SDLC?**
  - Meet expectations.
  - Code to operational software
  - Quality
  - What to do , when, by whom

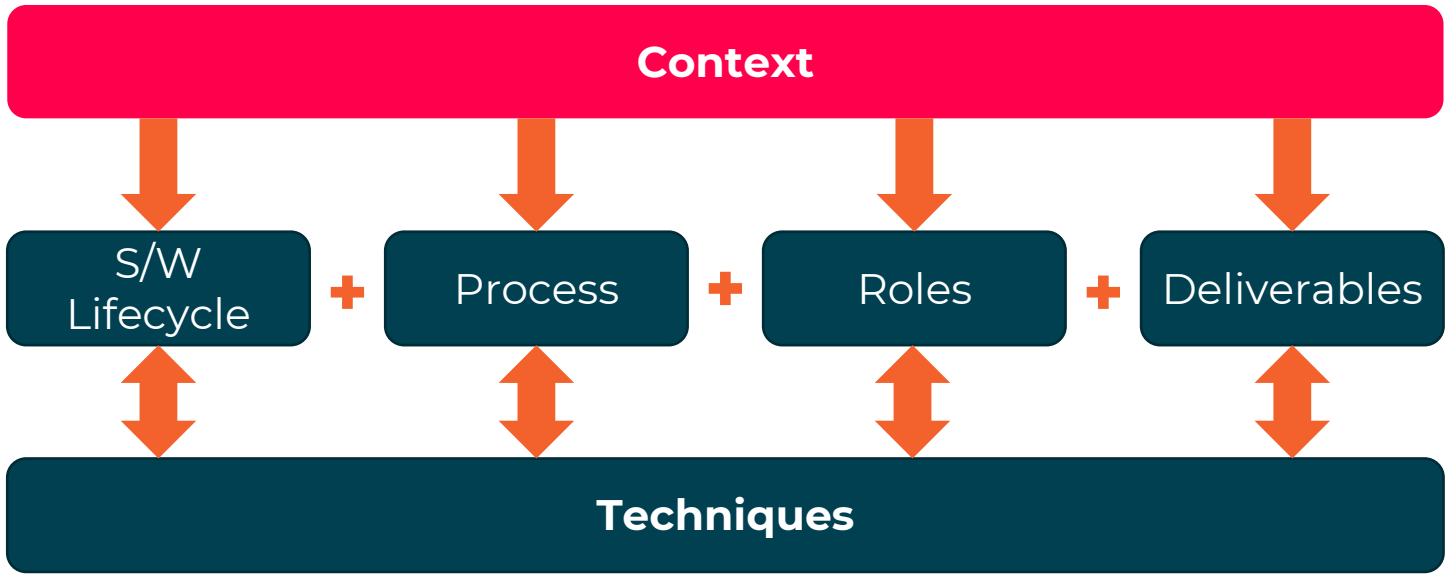
The cost of software failure

Elements of a framework method



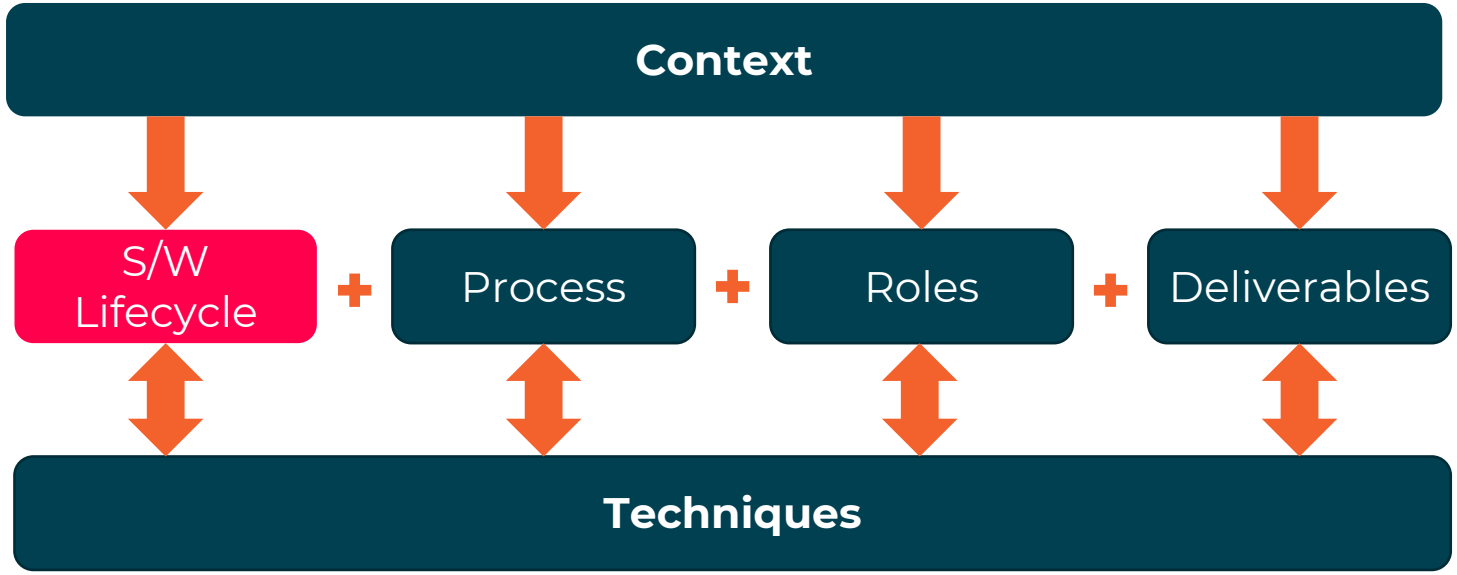
rules

Context



What must be considered before starting?	
Release plan	Single or multiple
Staff preferred delivery style	Level of skills and expertise
Location of teams and stakeholders	Single site or dispersed
Requirements stability	Audit, quality, regulatory, complexity and stability
Technology to be used	Tried and tested or new?

Lifecycle



Describes stages to follow: **Plan -> Design -> Build -> Test -> Deliver**

Approaches

Linear – Sequential, or step-by-step

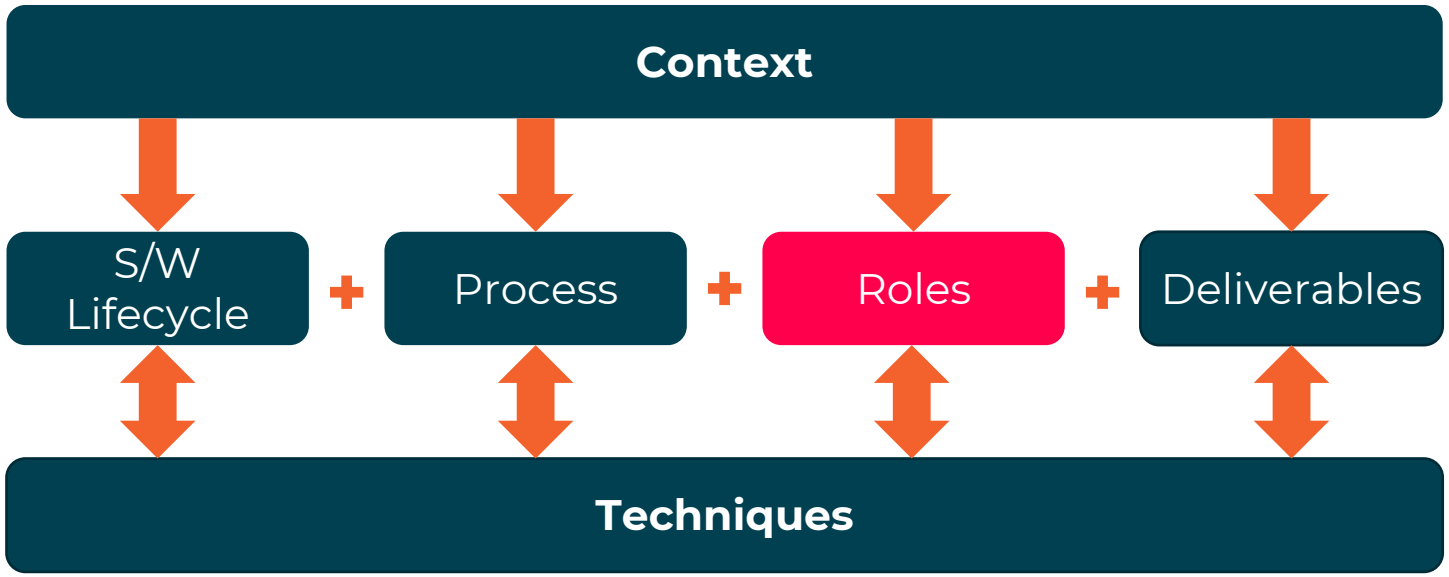
Evolutionary – Iterative evolving through versions

Prescriptive – **Specific SDLCs**

Agnostic – **Various SDLCs**

Dependent on context and lifecycle chosen

Roles



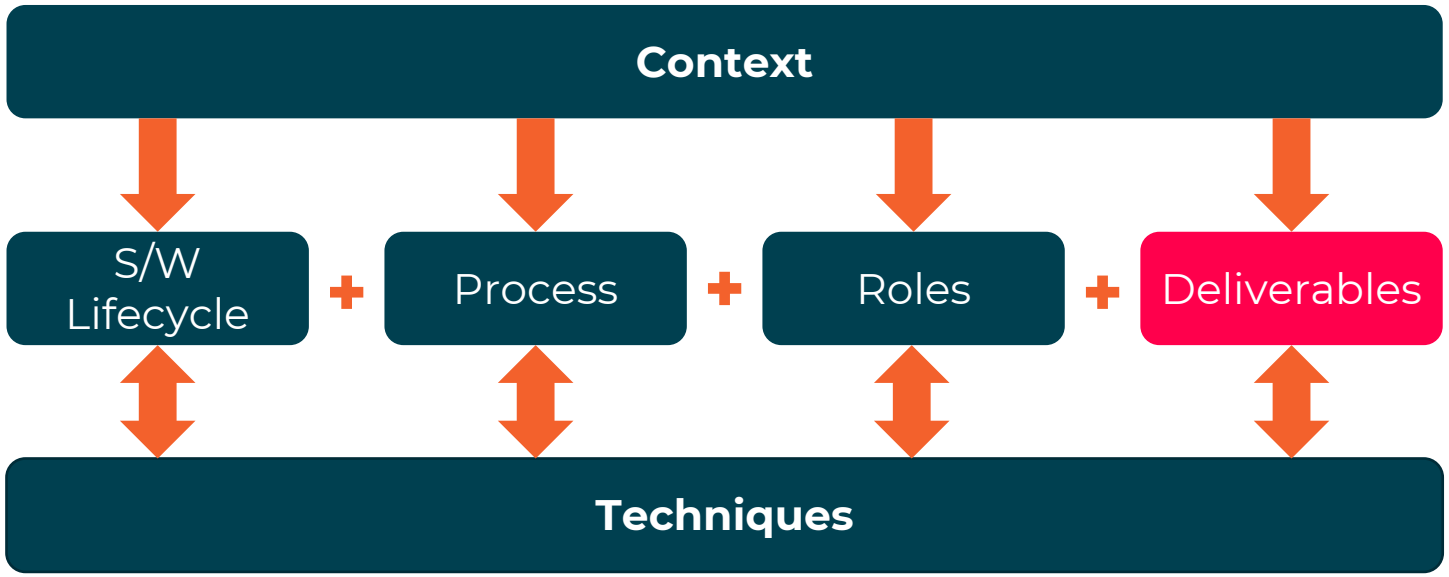
Implementation and Support:
Release Manager
Database Administrators
System Administrators

People that carry out tasks within various SDLCs

Specific to particular SDLC or generic – Various titles

Include Business, Project, Technical and Support roles

Deliverables



Models:

Class models
ERD, Data models or Logical Data structures
UML Use cases
Process models
State transition diagrams
Sequence diagrams
Component diagrams
System / software

Test plans
Deployment plans and scripts

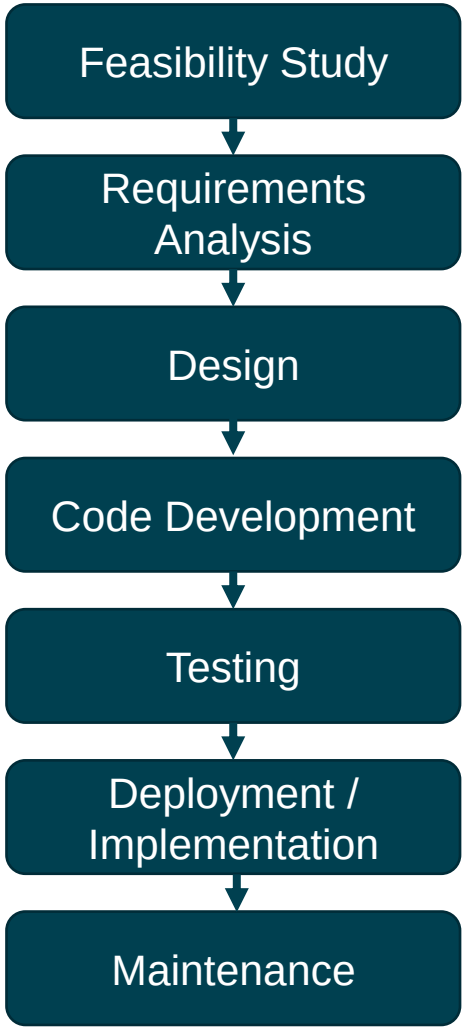
System Development Life Cycle (SDLC)

Is a framework describing a process to…

- Understand
- Plan
- Build
- Test
- Deploy

Can apply to

- Hardware
- Software
- Both



Requirement Analysis

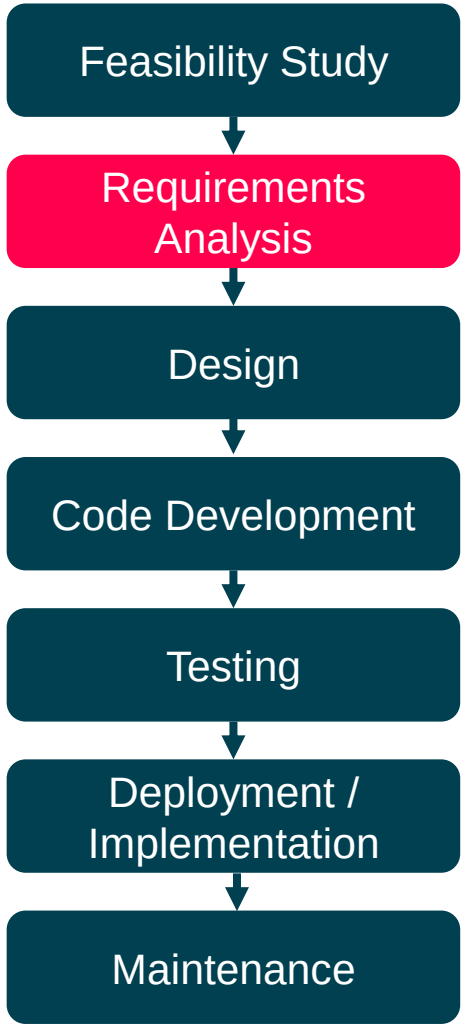
Gain understanding of what the business needs the proposed system to do
Business Analysts elicit, analyse, document, and validate requirements

Decide how to store, manage, access, and update requirements

Also known as Requirements Engineering (RE)

Make use of tools such as UML

What make a good requirement?
And why is it important?

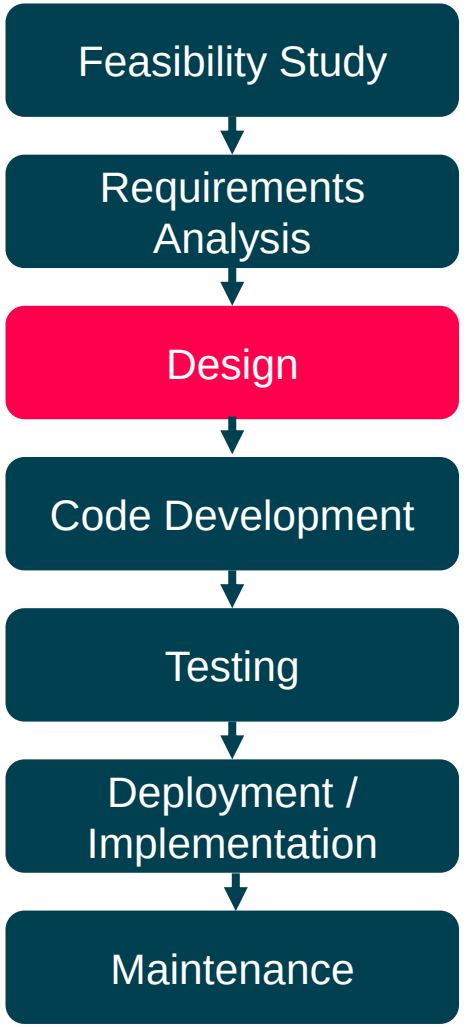


Design

Evaluate solutions that meet requirements.

Develop the chosen design with detail to begin development

Make use of tools such as UML



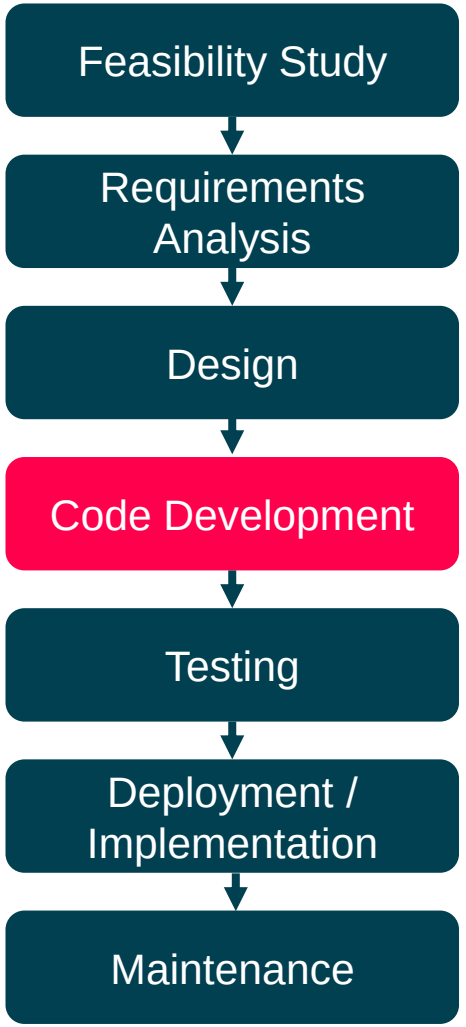
Code Development (**Programming or Build**)

Hardware and software technical components created, procured or configured

Follow design to ensure system does what is required

Make use of tools such as

- automated build tools
- code coverage
- testing frameworks



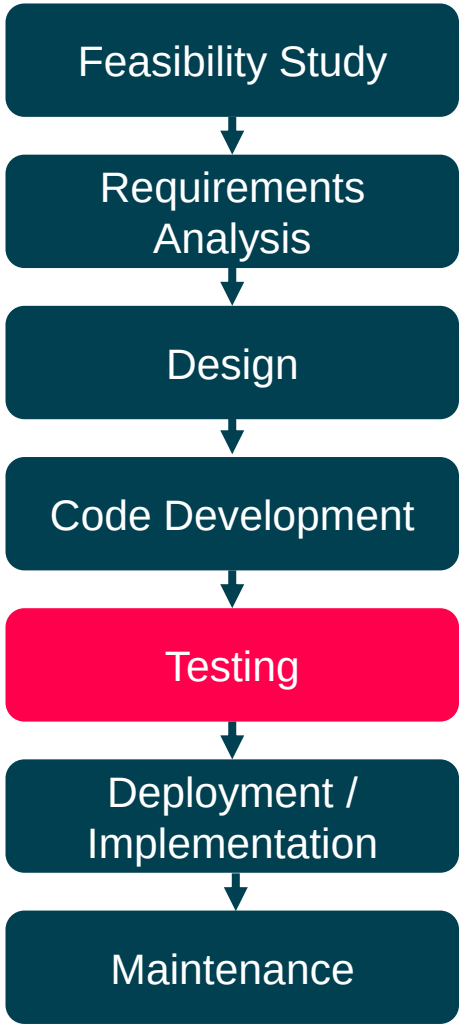
Testing

Components produced tested to ensure working properly and does what supposed to do

Different levels of testing

- Unit
- Integration
- System
- User Acceptance

Make use of tools such as automated build tools and code coverage and testing frameworks



Deployment / Implementation

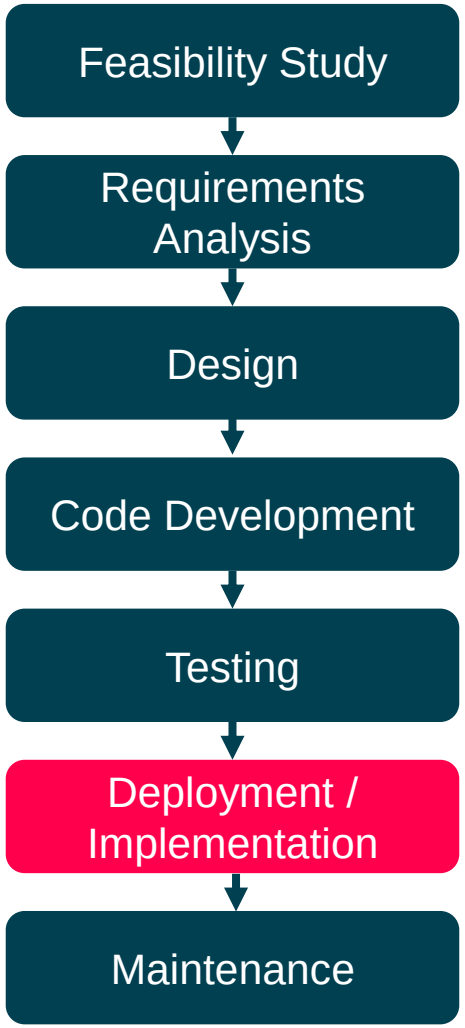
Commission the system in 'live' environment

- Known as **operation** or **production** environment
- Developed in 'test' environment

Must be carefully planned, understood and managed

Make use of tools such as

- automated build tools
- code coverage
- testing frameworks



Maintenance

More than bug fixing or keeping the system in good order

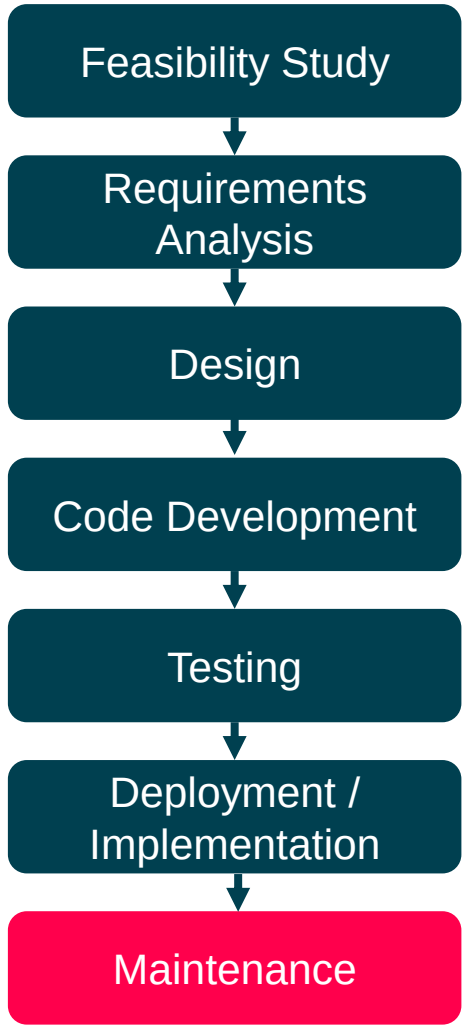
- **20%** Corrective
- **80%** Enhancements

Depends on

- Development strategy (one release or incremental)
- Lifetime of operation

Maintenance Types

- **Corrective** Fixing technical or requirement-based faults.
- **Adaptive** Making changes to meet new or updated requirements.
- **Perfective** Improve non-functional requirements
- **Preventative** Addressing foreseeable or hidden errors.



POWERING POTENTIAL

WHAT IS AGILE?

OBJECTIVES OF THIS SESSION

Understand What Agile is

Why you would want to use Agile for your Software Development Project

Appreciate the concept of the Agile Manifesto

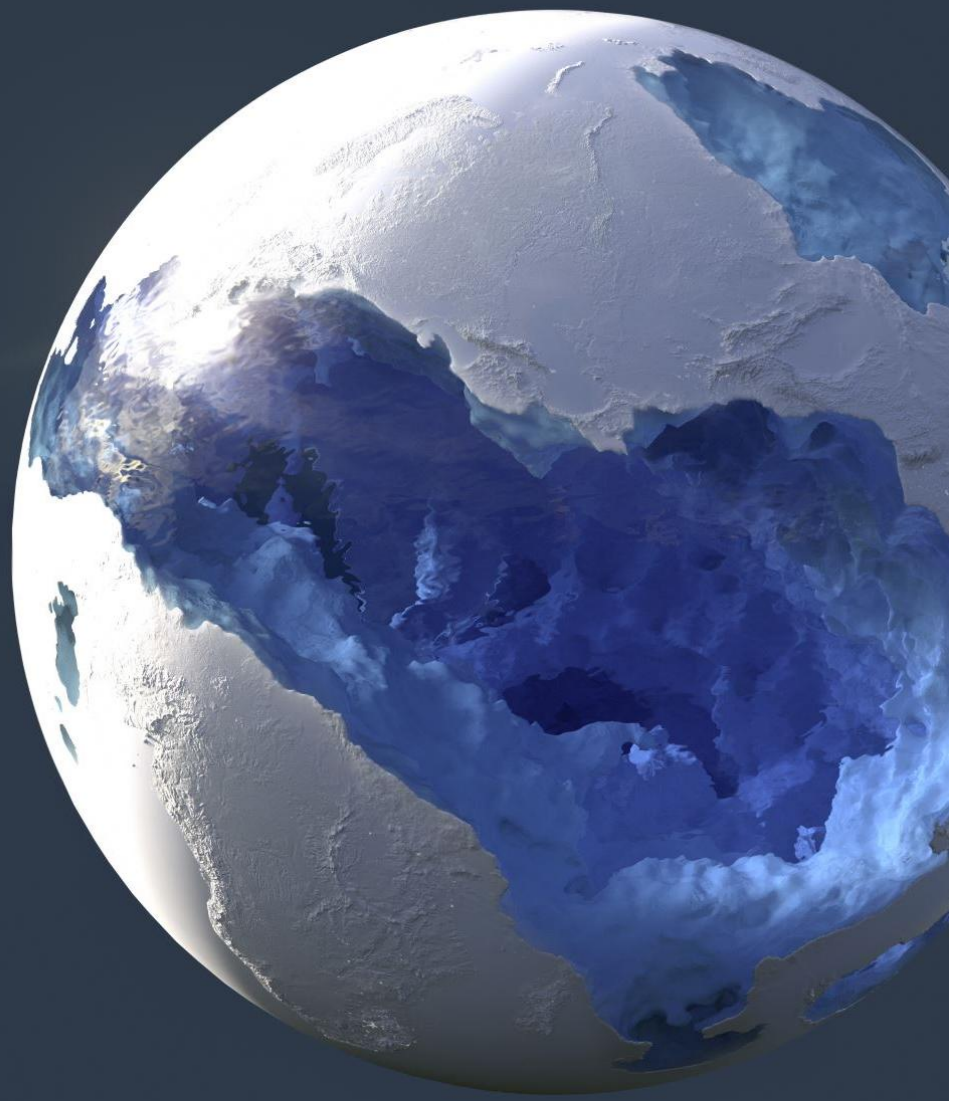
Understand and use the 12 Agile Principles to support the Agile Manifesto and the Agile Framework.

BEFORE AGILE

WATERFALL

- **Tons of documentation up front**
  - Business requirements, Application's architecture
  - Data structures, Functional designs
  - User interfaces, non-functional requirements
- **No code before all design are complete**
  - Then Tests and eventually deploy
- **Large teams needed, even for a small project**
- **But it worked!**
  - Systems were large, monolithic, clear outcome
  - Requirements changed slowly

- **We now require speed, flexibility in a changing world**



Why Agile?

The dynamic and fast-moving nature of the world today:

- **V**olatility
- **U**ncertainty
- **C**omplexity
- **A**mbiguity

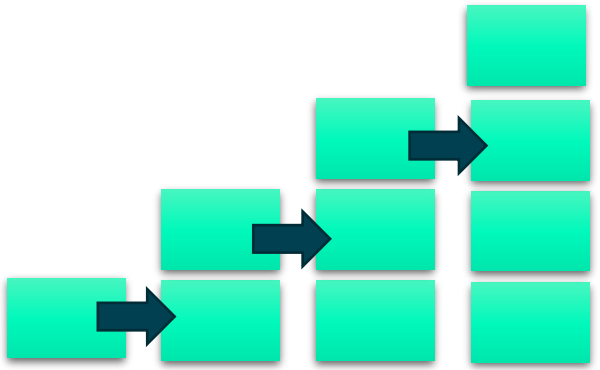
In this environment, the assumption that a solution can be designed in detail up-front does not reflect reality.

WHAT IS AGILE

Agile is the most common software development methodology

It is an evolutionary methodology

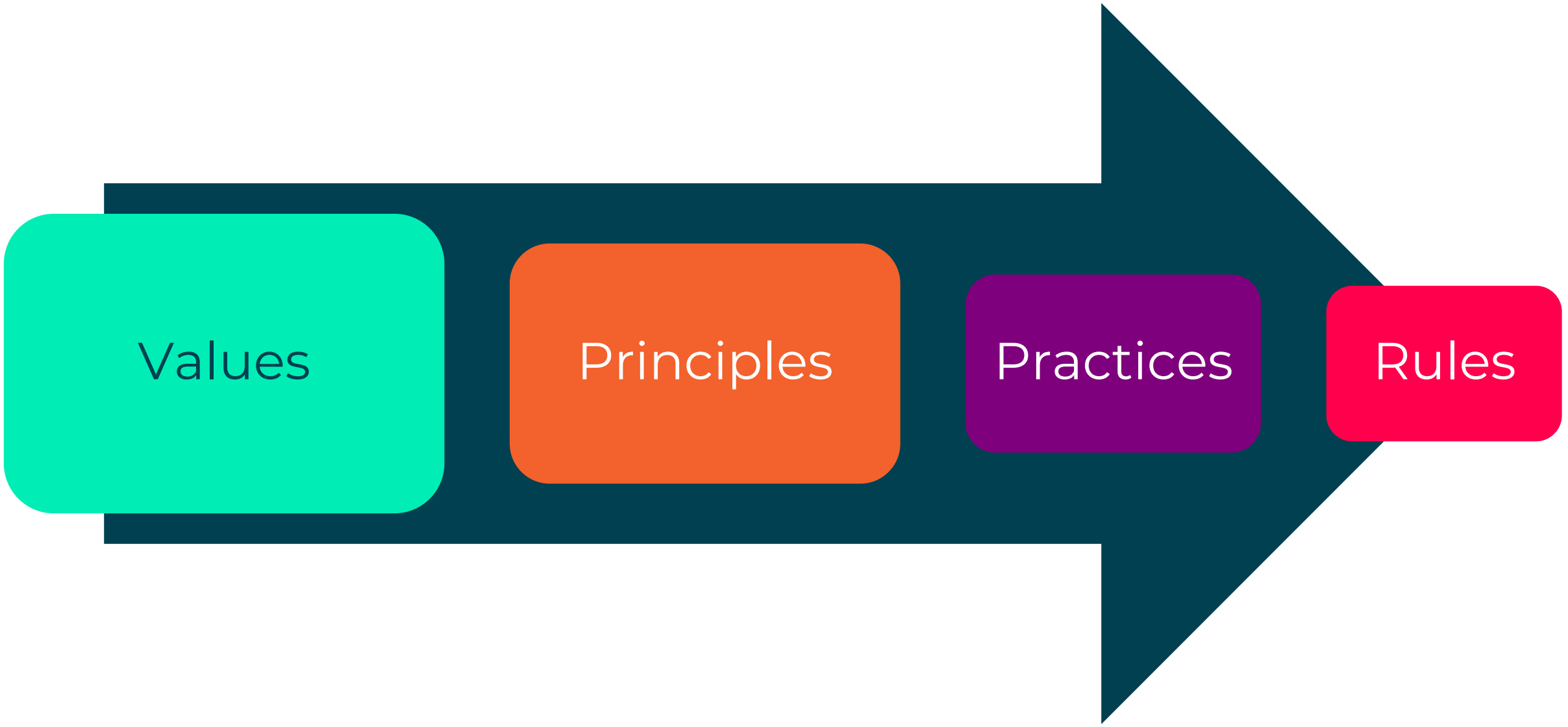
The software solution is developed through early prototype or iteration releases



EVOLUTIONARY AND WATERFALL COMPARISON

by blog.fastmonkeys.com

What is important in Agile?



Why USE Agile?

Requirements not well understood

- Difficult to define and express what is needed
- Iterations uncover understanding
- Demo working software to gather feedback

Early delivery more important than completeness

- Early benefits – Justify funding £
- Quick to market – High value features first

High business/technical risk.

AGILE

Launched in **2001**

17 technologists drafted the **Agile Manifesto**

Devised 4 principles for agile software production

THE AGILE MANIFESTO

A framework is Agile if it follows the Agile Manifesto.

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

AGILE PRINCIPLES 1-6

We follow these principles:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale
4. Business people and developers must work together daily throughout the project
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

AGILE PRINCIPLES 7 - 12

7. Working software is the primary measure of progress
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely
9. Continuous attention to technical excellence and good design enhances agility
10. Simplicity – the art of maximising the amount of work not done – is essential
11. The best architectures, requirements, and designs emerge from self-organising teams
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly

What Agile is about?

Discussion over documentation

Small steps done often

Encourages change

People act autonomously

No silo mentality

Welcome reviews

Encourage teamwork

Trust people to do their job

Communicate best practices

Fail fast and adjust quickly!

Regular demonstration of results

AGILE SCRUM

The Scrum framework gets its name from Rugby
In Rugby, the individual players work as a single unit to move.

- Is repetitive and will produce a runnable result at the end of every sprint which is ‘shippable’; not half-broken
- The code will add functionality to what was there previously, therefore adding value.

PRODUCT BACKLOG

Is the list of all features, enhancements and fixes to be made to the product in future releases.

Items include a description, order, estimate, and value.

The **product owner** is responsible for the product backlog, its content, availability, and ordering.

The dev team collaborates with the product owner, adding details to the backlog items where necessary

DEFINITION OF READY (DOR)

Defines what a PBI needs before it can go into the sprint backlog.

A checklist of items could include:

Technical details discussed & agreed

Priority assigned

Acceptance Criteria defined

Product Backlog Item (PBI)

Must have a **clear description**.
A clear user story

Offer a **Business value** properly articulated and agreed

Acceptance criteria is clear and **testable**

Must have the **staff** to do it

dependencies on other systems/things are identified.

The Product Owner has **approved** it

DEFINITION OF DONE (DOD)

Definition of done (DoD)
defines what is needed before it can be regarded as complete. Can be applied to a feature, a Sprint or a release.

A checklist of items which could be included for a feature could include:

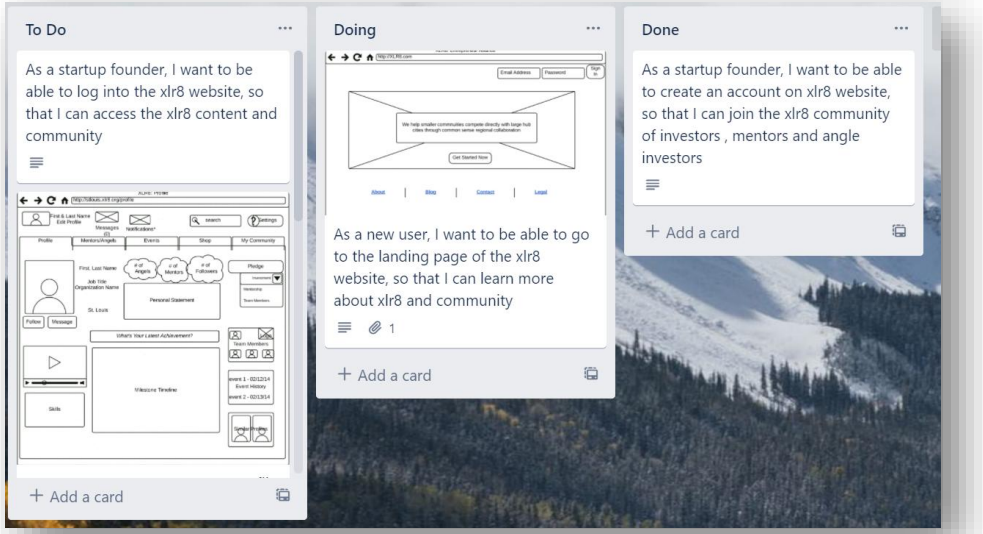
- Unit testing written and passed
- Documentation updated
- Peer code review completed

SPRINT BACKLOG

A sprint backlog is usually presented in a Kanban board.

As tasks are worked, the cards move from one end of the board to the other. This movement helps the team to see the project's progress.

Trello is a common project tracking tool.



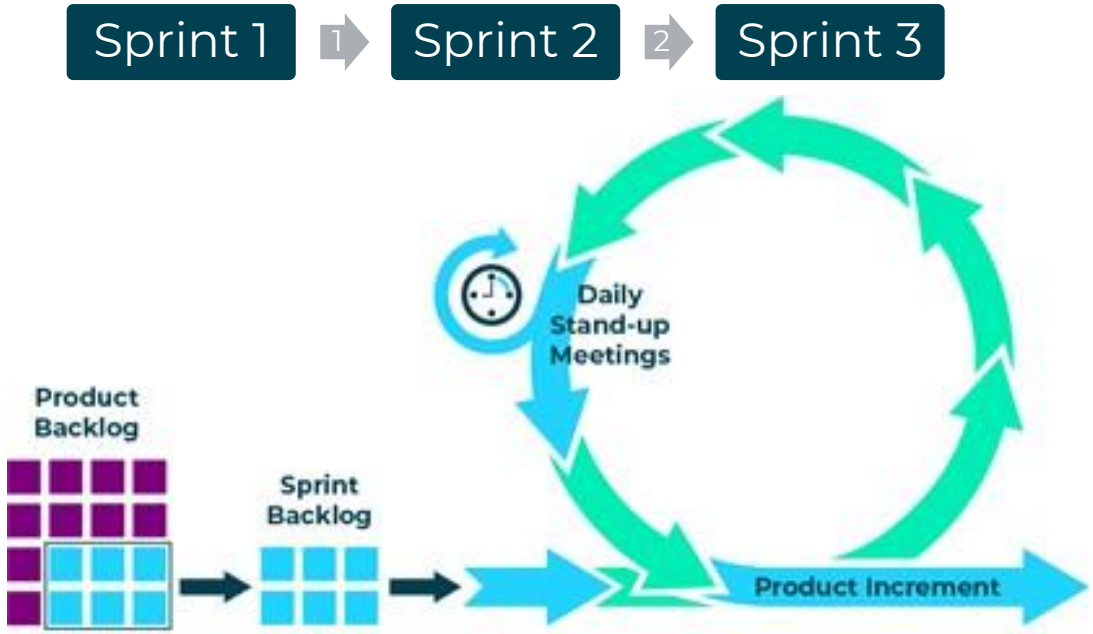
WHAT IS A SPRINT?

A sprint is **a time-boxed** event

The aim is to produce new production-ready code

A sprint can be between 1 to 4 weeks long

Sprints will run one after another until the product no longer needs development.



ROLES

Roles which people take in an Agile Project

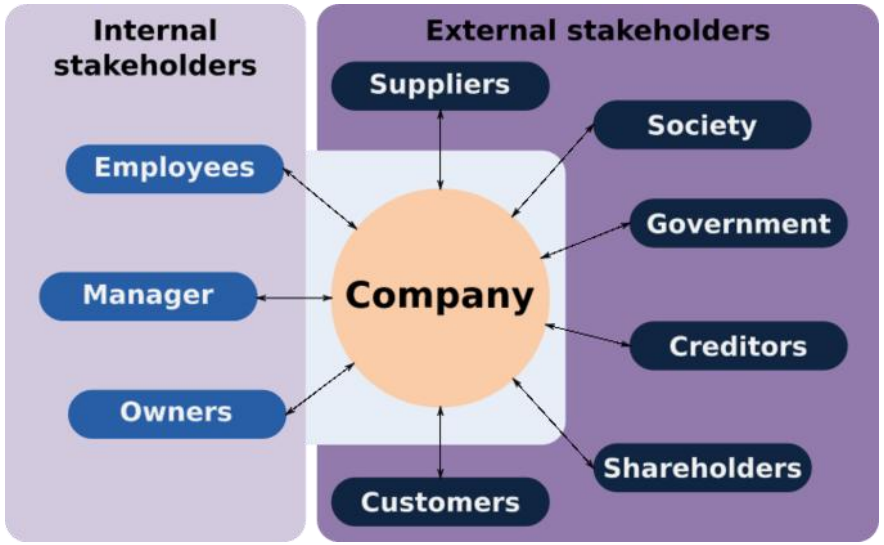
- Stakeholder
- Project manager
- Product Owner
- Business Analyst
- Scrum Master
- Development Team

THE STAKEHOLDER

A stakeholder is anyone with an interest in or an influence on the product.

Should be contactable by the development team to help them to understand what is required.

They should attend the Sprint review meeting



PRODUCT OWNER

Is the sole person responsible for managing the product backlog (list of features to be worked on).

They communicate with the stakeholders to ensure requires are added to the product backlog

They attend the Sprint planning meeting and Sprint review meeting

PROJECT MANAGER

Manages projects and will overview expenses.

Try to reduce risk on the project.

Can also manage more than one project at a time.

BUSINESS ANALYST (BA)

Supports the product owner by gathering requirements and providing guidance on what to build.

They usually work across many products.

SCRUM MASTER

Helps those outside the scrum team understand which interactions are beneficial.

Help find techniques with effective Product Goal definition and Product Backlog management

Supports the development team by removing impediments, facilitating meetings and coaching self-organisation

Attends the Sprint planning, Daily stand up, Sprint review and the Sprint Retrospective meetings

DEVELOPMENT TEAM

A multi-disciplinary development team usually consists of software architects, designers, programmers and testers.

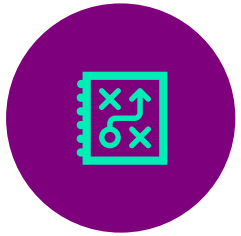
This is a self-organising team.
They decide how to tackle the items in the sprint backlog.

People with other skillsets can be added to the team

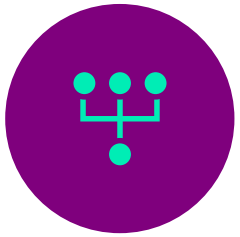
Attend the Sprint planning, Daily stand up,
Sprint review and the Sprint Retrospective meetings

Sprint planning meeting

Held at the beginning of the sprint.

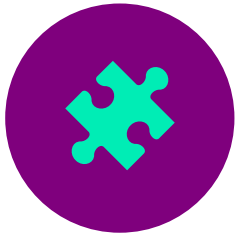


May take a day for a two-week sprint and is split into two sections (what and how).



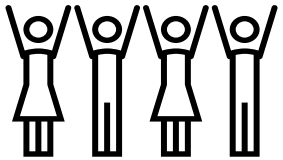
Team decide **what** is going to be brought into the sprint backlog.

Team decide **how** the items are going to be completed,



Done by breaking down the tasks into smaller pieces and adding technical details.

DAILY STAND UP



This is a daily 15-minute meeting used to optimise communication across the team.

Following points are discussed by every participant:

- What did I do **yesterday** that helped meet the sprint goal?
- What will I do **today** to help meet the sprint goal?
- Do I see any future **impediments** that prevents the team from meeting the sprint goal?

SPRINT REVIEW MEETING

Held at the end of the sprint to inspect the work done and adapt the product backlog.

- The dev team demonstrates what work was done, and answers any questions
- The product owner discusses the product backlog as it stands
- The entire group collaborates on what to do next, to provide valuable input to future sprint planning

The dev team, product owner and key stakeholders attend this meeting.

SPRINT RETROSPECTIVE MEETING

Is an opportunity for the scrum team to inspect itself and create a plan for improvements to be made during the next sprint.

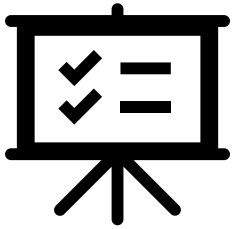
The goal is to:

- Inspect how the last sprint went with regards to people, relationships, process, and tools
- Identify what went well and any potential improvements that could be made
- Create a plan for implementing improvements to the way the scrum team works

USER STORIES

Let’s examine the role of User Stories in an Agile Project

- User story format
- Acceptance Criteria



USER STORY

Is a simple description of a product feature that is written from an end user's viewpoint

User stories consist of three parts

1. **Who** is the user / customer

2. **What** do they want to do

3. **Why** do they want to do it (value to the user)

WRITING USER STORIES

As a (user), I want (goal) so that I can (value or why)

The user story is:-

1. **Independent** of other user stories
2. **Feasible**. Can change and adapt to the users' needs
3. **Valuable**. It has concrete value to the user
4. **Estimable** how complex is it
5. **Small** enough for a team member to finish in time
6. **Testable**

EXAMPLE

User Story 1 –
Create an account for a new user

As a subscriber

I want to be able to create an account on my technical website

So that I can learn the best technical indicators to help me improve my trading.

This user story requires further criteria so that a developer can start work

ACCEPTANCE CRITERIA

Acceptance criteria dictates the conditions for software to be considered done.

It is a set of statements that usually have a pass / fail result for all requirements.

Attached to user stories to understand what a feature needs.

Defines the minimum viable product.
Can also derive tests from the criteria.

EXAMPLE

User Story 1 - Create an account for a new user

As a **subscriber**

I want to **be able to create an account on MyTechnicals website**

So that **I can learn the best technical indicators to help me improve my trading.**

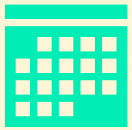
Acceptance Criteria

1. **Able to create an account manually (filling out the sign-up form)**
2. **Able to create an account via Facebook**
3. **Able to create an account via Google**
4. **Able to create an account via LinkedIn**

The user story is now DoR

Time estimation - Story Points NOT Time

Traditionally when prioritising and scheduling tasks, analysts would use Complexity and Time as key factors



This has proven to be a unsatisfactory way of prioritising and scheduling workloads

Recommended approach – **story points**

A value assigned that represents its complexity and time to deliver

Estimation

Techniques for assigning story points

- Estimation Poker - https://www.planningpoker.com/

https://agilescrumgroup.nl/scrum-planning-poker-swimlane-sizing/

SUMMARY

What Agile Scrum is

What the product backlog

Definition of Ready (DoR)

What is the sprint backlog

Definition of Done (DoD)