## Introduction to Agile

* 1. Software development as a complex adaptive system.
  2. Waterfall – main issues and root causes

**The waterfall (cascade) model**

The “classic” project management model.

The project is a liner one. There are no iterations.

Each part of the project is an individual set of actions.

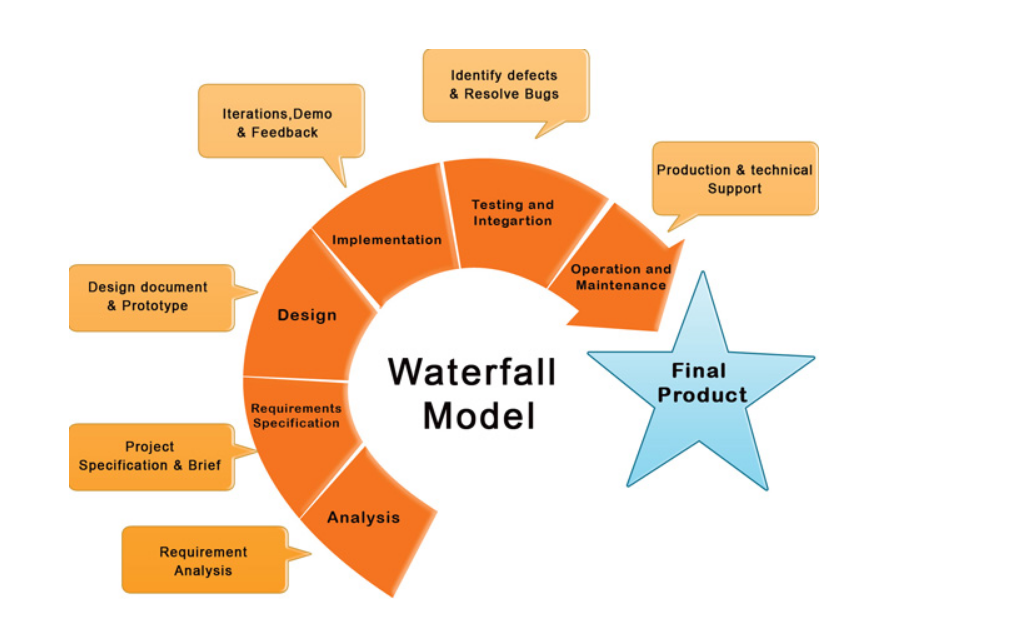
One starts when the previous is over

Original roots in production, engineering and construction. It is later adopted in SW development.

Widely used by large corporations, governmental projects.

Easy to manage, the deadlines are strict and set before a start of the project.

The product development is often slow, as most times the final product is somehow different from the initial expectations.



**Waterfall is not all bad**

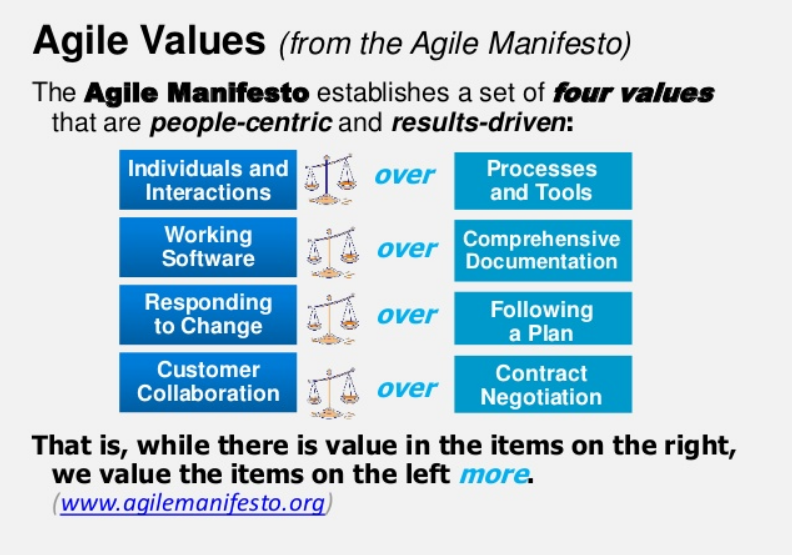
Waterfall is not without sense.

Original code was written on paper.

The model seeks risk reduction through early planning.

Most projects were developed for large scale organizations.

* 1. Agile – main advantages over Waterfall
  2. The Manifesto for Agile Software Development – goal, values, principles



* 1. Modern Agile – core tenets and what they mean

“Agile is a **set of methods and frameworks** that embody the principles and values of the “**Agile Manifesto**”**.**

“Agile is a term used to describe **approaches to software development** emphasizing **incremental delivery, team collaboration, continual planning, and continual learning**. The term “Agile” was coined in 2001 in the “**Agile Manifesto**”.

*“*Being ‘Agile Is a **mindset**. It’s about finding the right thing to build, faster (and not just building things faster)”

*“*Agile is a time boxed, **iterative approach** to software delivery that builds software **incrementally** from the start of the project, **instead of trying to deliver it all at once near the end**.”

*“*The ability to create and **respond to change** in order to succeed in an uncertain and turbulent environment.”

*“*Agile is best suited for situations that have some **level of uncertainty** where **creativity** and **innovation** are important to **maximize the business value** of the solution as opposed to other situations with lower levels of uncertainty where planning and control to achieve predictability are more important*.”*



Agile is a Mindset.

It is build on Values, Principles and Practices.

It is developed for SW development, but applicable outside of it.

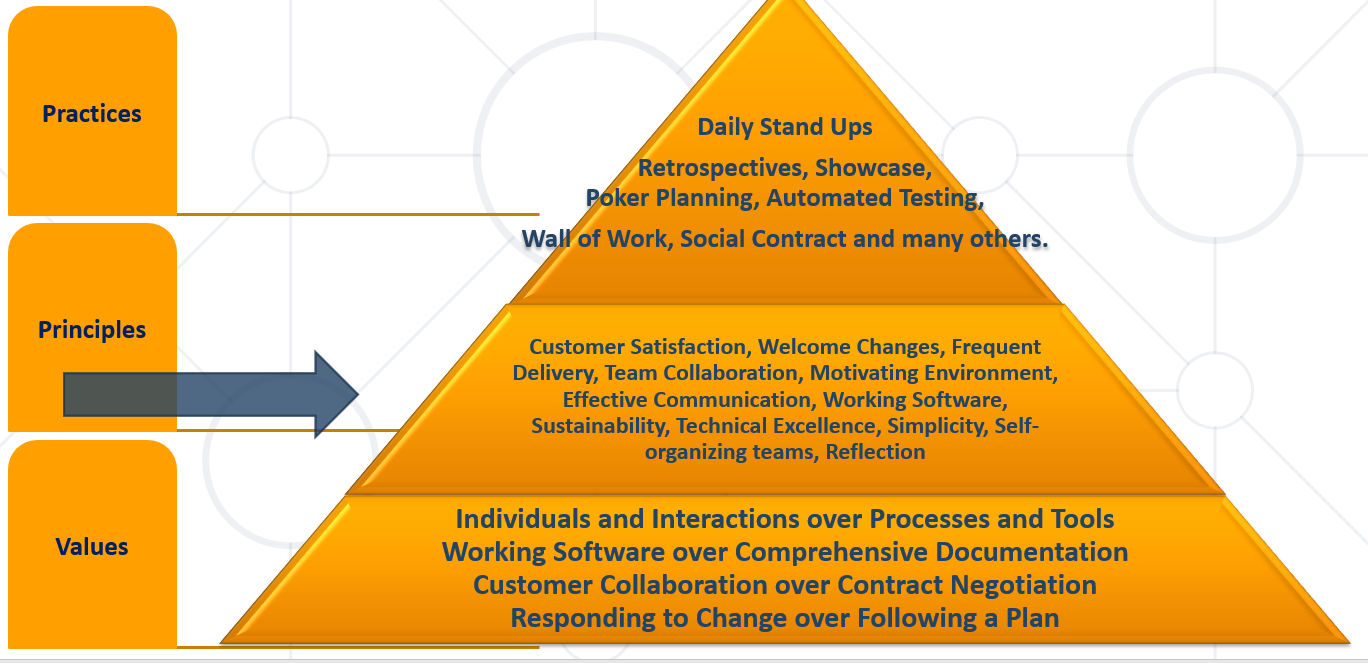
It is developed by practitioners.

It was built on top of existing methodologies.

And it is still evolving.

Agile is NOT a project management methodology.

Agile is NOT an answer to every problem.



## Agile in Practice

* 1. Teams – what makes a good team for Agile software development?

Product Owner

Agile TL / Scrum Master

Development Team

Business Analyst

UI/UX

Customer (not really part of the team, but the important part of the process)

**Best Practices**

* Coding Standards
* (Peer) Code Review
* Pair Programming
* Test-driven Development (TDD)
* Automated Tests
* Continuous Integration and Continuous Delivery
  1. Planning – how do we ensure realistic commitments and achievable plan

**Building the roadmap**

*A product roadmap is a plan of action for how a product or solution will evolve over time. Product owners use roadmaps to outline future product functionality and when new features will be released. When used in agile development, a roadmap provides crucial context for the team's everyday work and should be responsive to shifts in the competitive landscape.*

**Release Plan**

Built around short-term sprints with the goal of regularly and consistently releasing usable software

High-level timetable for each release

Feature prioritization by release

Agile project plan is divided into releases and iterations

User Stories

Acceptance Criteria

* 1. Iterations and flow – main differences

**Iteration Planning**

Macro to micro view

Iteration backlog

Iteration goals

Iteration tasks

**Iteration Tasks**

* Create a task board
* Assign tasks to team members
* Clear definition
* Timebox
* Progress tracking

**Evaluation and Success**

Key elements of success of the sprint:

* Stand Up
* Effective Communication
* Measurement

*What did I do yesterday?*

*What will I do today?*

*Do I have any problems or concerns?*

* 1. Feedback – importance of, examples of, frequency

**Iteration Review and Demo**

Work done during the iteration is discussed

Finished increment is shown to the customer or the stakeholders

Collecting feedback from everyone:

Achievements

Impediments

Challenges

Retrospective session:

What was done well?

What went wrong?

What to improve?

## Agile Product Management

* 1. Software Economics – understand the four laws
  2. Knowing your users – personas, experiments, dogfooding, early involvement, adoption curve
  3. Focusing on working software – definition of done, technical debt
  4. User stories – essence, purpose, what they are not,

What is a User Story?

Short, simple descriptions of an objective.

The story is always told from user’s point of view.

It should be doable in a single iteration.

If the “story” can be fulfilled in a single iteration, or it is too broad, then it should be broken into smaller, simpler “stories”.

A bigger “story” is also known as a “**Epic**”.

Usually, the story would always follow the same format:

As a (*the typical user of the product/service*) …

I want to (*achieve a certain goal*)…

Because (*the desired outcome*)…

*So I am able to…(another format)*

*At the end of the day think of: WHO, WHAT and WHY.*

A “**Project Objective**”, also defined as a “**Hill**” is a general objective which guides the team and helps define the final objective.

An “**Epic**” is a bigger story which usually can not be completed in a single iteration, or captures too many objectives, hence it should be broken down to smaller pieces.

An “**User Story**” is a tangible product item which can be worked by the team in a single iteration. At the end of the iteration, the “User Story” should be: testable, able to be presented and executable.

* So who is creating the “User Stories”?
  + Product Manager – but would he always be able to deliver a workable “user story”, or an “Epic”?
  + The “Delivery Team”
* To create an User Story, one can use a specific SW, predefined template, or simply sticky notes.

**Defining Acceptance Criteria**

* Each User Story will have associated User Story Acceptance Criteria (also referred to as “Acceptance Criteria”).
* Used to articulate when exactly the User Story is done from Product Owner perspective.
* At the end of each Sprint, the Product Owner uses these criteria to verify the completed deliverables. Can either accept or reject individual deliverables and their associated User Stories.

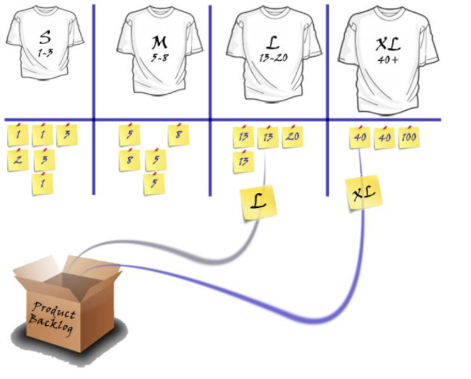
**Definition of Done**

* While Acceptance Criteria are unique for individual User Stories, Done Criteria are a set of rules that are applicable to all User Stories in a given Sprint.
* As with the Acceptance Criteria, all conditions of the Done Criteria must be satisfied for the User Story to be considered Done.
* A good practice is to use a checklist to verify if all criteria are met.
* A clear Definition of Done is critical.
* General Done Criteria could include any of the following:
  + Reviewed by other team members
  + Completed unit testing of the User Story
  + Completion of quality assurance tests
  + Completion of all documentation related to the User Story
  + All issues are fixed
  + Successful demonstration to stakeholders and/or business representatives
  1. Other forms of work – features, spikes, etc.

A **spike** is a **user story** for which the team cannot estimate the effort needed. In such a case, it is better to run time-boxed research, exploration to learn about the issue or the possible solutions. As a result of the **spike**, the team can break down the **features** into **stories** and estimate them.

* 1. The Product Backlog – what makes a good backlog
  2. Story Mapping – how it works, benefits over a flat backlog
  3. Slicing user stories – purpose, techniques, success criteria
  4. Managing multiple backlogs – when and why would we organize backlogs a certain way?

## Estimation, Planning and Tracking

* 1. Targets, estimates, commitments
  2. The Cone of uncertainty – the effect of planning horizon distance on estimation accuracy
  3. Story points – what they represent and what they are not. Key properties – subjectivity, affinity
     1. Components of size – effort, complexity, risk
  4. Fixed scales – why use them. The Fibonacci scale.
  5. Ideal days – what they are and what they are not. Difference with Story Points.
  6. Estimation rituals: planning poker, reference items, buckets. What to do, what not to do. Main benefits and drawbacks of.

Speed over accuracy

Collaborative techniques

Relative units

T-shirt size system: XS, S, M, L, XL, XXL

* 1. Re-estimation – when to do it and when not to do it
  2. Overestimation vs. underestimation
  3. Tracking
     1. Burndown charts – work done, scope added, projections
     2. Cumulative flow diagrams – cycle time, throughput, projections, bottlenecks

## Inspect, Adapt, Retrospect

* 1. The importance of feedback in a complex adaptive system
  2. Effects of feedback loop length
  3. Nesting feedback loops – benefits, examples
  4. Other forms of feedback
  5. Retrospectives – essence, the Prime Directive, essence, stages
  6. Good vs. bad retrospectives

## Agile Metrics

* 1. Rationale – why use metrics, key benefits and pitfalls
  2. Metrics, Insights, Decisions and Outcomes
  3. Metrics quality – what makes a good vs. bad metric
  4. Dimensionality and balance
  5. The six main dimensions of agility
  6. Key metrics: throughput, velocity, cycle time, lead time, queue length, predictability
  7. Insights – trends, correlations, cause-effect
  8. Decisions – teams, processes, practices
  9. Outcomes – real life examples (team size, WIP size, branch lifetime)

## TPS, Lean, Kanban

The Toyota Production System (TPS) is the cornerstone of Lean Management. It is best described as: “an Operations Management System to achieve goals of the highest quality, lowest cost and the shortest lead-time via engaging people towards goals.”

* 1. Overburden and inconsistency
  2. Types of waste. How do they translate to software development?
  3. Just-in-Time decision making. Examples in Agile software development?
  4. Intelligent automation – examples in software development? (hint: CI/CD, DevOps)
  5. Continuous improvement, respect for people – Agile perspective?
  6. Quality and integrity. Examples in software development.
  7. Local (sub)optimization – examples, dangers
  8. Value streams – value-adding activities, waste
  9. Continuous flow. Pull systems. WIP limits
  10. Kanban – boards (columns, lanes, WIP limits, policies), flow (priority of execution)

Developed by Toyota for **lean** manufacturing and **just-in-time** manufacturing.

Focus on improvement.

Visualize work.

A signal tells a supplier to produce and deliver a new shipment when a material is consumed.

Ohno is also known for his "Ten Precepts" to think and act to win.[[7]](https://en.wikipedia.org/wiki/Taiichi_Ohno)

You are a cost. First reduce waste.

First say, "I can do it." And try before everything.

The workplace is a teacher. You can find answers only in the workplace.

Do anything immediately. Starting something right now is the only way to win.

Once you start something, persevere with it. Do not give up until you finish it.

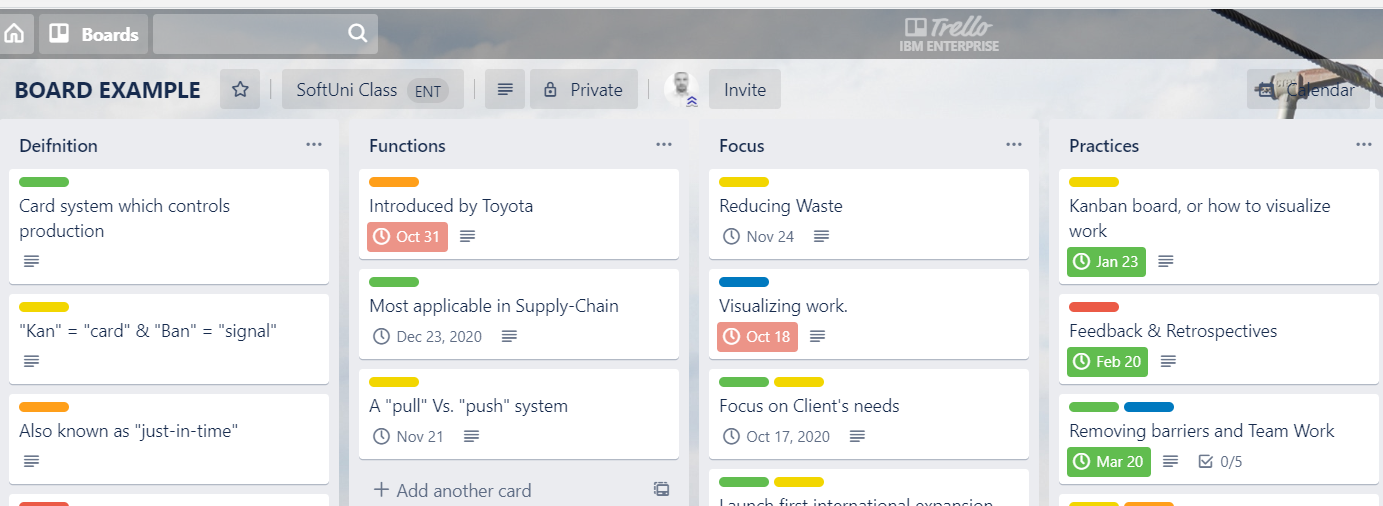
Explain difficult things in an easy-to-understand manner. Repeat things that are easy to understand.

Waste is hidden. Do not hide it. Make problems visible.

Valueless motions are equal to shortening one's life.

Re-improve what was improved for further improvement.

Wisdom is given equally to everybody. The point is whether one can exercise it.



## Scaling Agile

* 1. Start with why. The right reason to consider Agile
  2. Big Bang vs. Iterative/Pilot approach
  3. Whom to involve?
  4. What we need to put in place?
  5. Prerequisites for Agile to even exist
  6. Common mistakes
  7. Common challenges (top 5) – culture, resistance, support, skills, training/coaching
  8. Success tips – training/coaching, consistency, sponsorship
  9. Scaling frameworks common grounds - number of teams, shared backlog, cadence, synchronization, shared events (planning, demo, retrospective)

1. Scrum – Plan, sprint, evaluate

Based on short cycles.

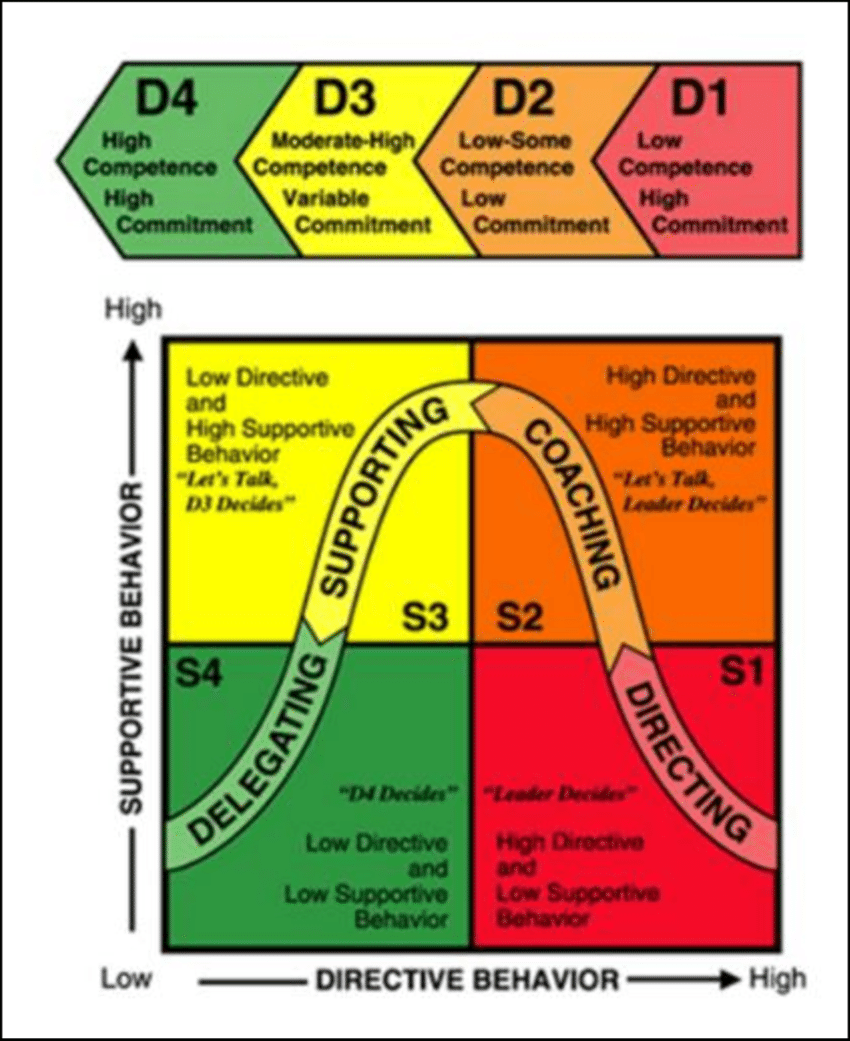
Typical cycle varies from 2 to 4 weeks.

After each cycle the team evaluates its strategy and adjust tactics.

It is almost a never ending circle which repeats.

At the end of the cycle there is always a workable product.

1. Situational leadership



A **timebox** is a previously agreed period of time during which a person or a team works steadily towards completion of some goal. Rather than allow work to continue until the goal is reached, and evaluating the time taken, the timebox approach consists of stopping work when the time limit is reached and evaluating what was accomplished.

An **iteration**, in the context of an Agile project, is a timebox during which development takes place, the duration of which:

may vary from project to project, usually between 1 and 4 weeks

is in most cases fixed for the duration of a given project

In consultation with the customer or product owner, the team divides up the work to be done into functional increments called “**user stories.”**

Each user story is expected to yield, once implemented, a contribution to the value of the overall product, irrespective of the order of implementation.



**Personas**

When the project calls for it – for instance when user experience is a major factor in project outcomes – the team crafts detailed, synthetic biographies of fictitious users of the future product: these are called “personas”. (In Alan Cooper’s concise terms: “make up pretend users and design for them”.)

**Backlog**

A product backlog is a list of the new features, changes to existing features, bug fixes, infrastructure changes or other activities that a team may deliver in order to achieve a specific outcome.

**Retrospective**

The team meets regularly, usually adhering to the rhythm of its iterations, to explicitly reflect on the most significant events to have occurred since the previous such meeting, and take decisions aiming at remediation or improvement.



The process of planning, creating, testing and deploying a software is called **Software Development Life Cycle** or **SDLC**.

The **Agile SDLC model** is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to four weeks. Every iteration involves cross- functional teams working simultaneously on the defined SDLC phases.

**Cross-functional Team**

* A cross-functional team has access to all the skills necessary to effectively deliver value to customers
* A cross-functional team does not have to consist of all generalists
* A cross-functional team also does not have to consist of all specialists
* Benefits of a cross-functional team:
  + Getting to done
  + Self-organizing
  + Innovative
  + Increased flow

People who develop skills outside of their chosen or preferred skillset, have become known as “T-shaped”

**Agile Roles and Responsibilities**

* Agile team lead (ScrumMaster)
* Product Owner
* Development Team
* Together all three roles make up an **Agile Team**