



Agile 4 Values



Working Software

Over

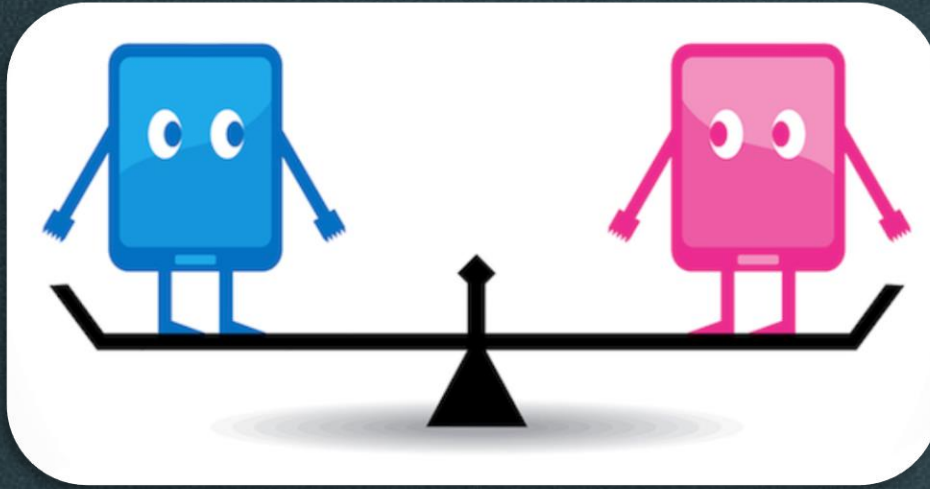
Comprehensive Documentation



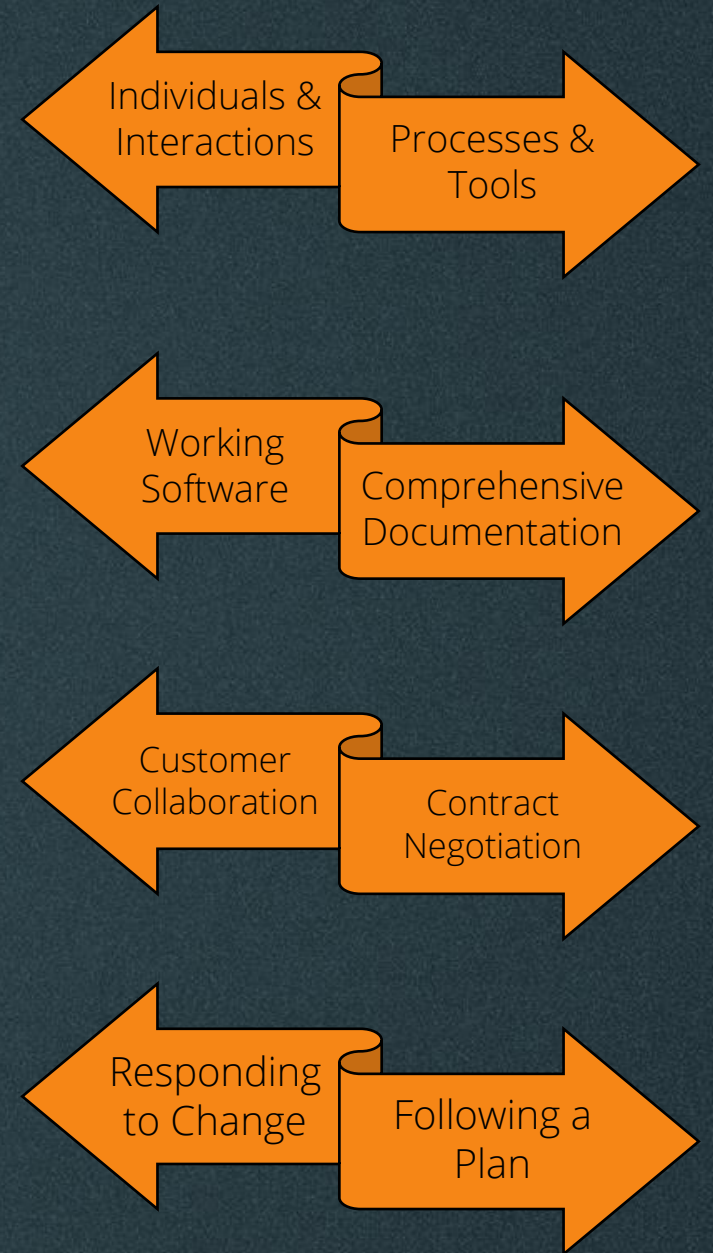
Customer Collaboration Over Contract Negotiation



Responding to Change Over Following a Plan

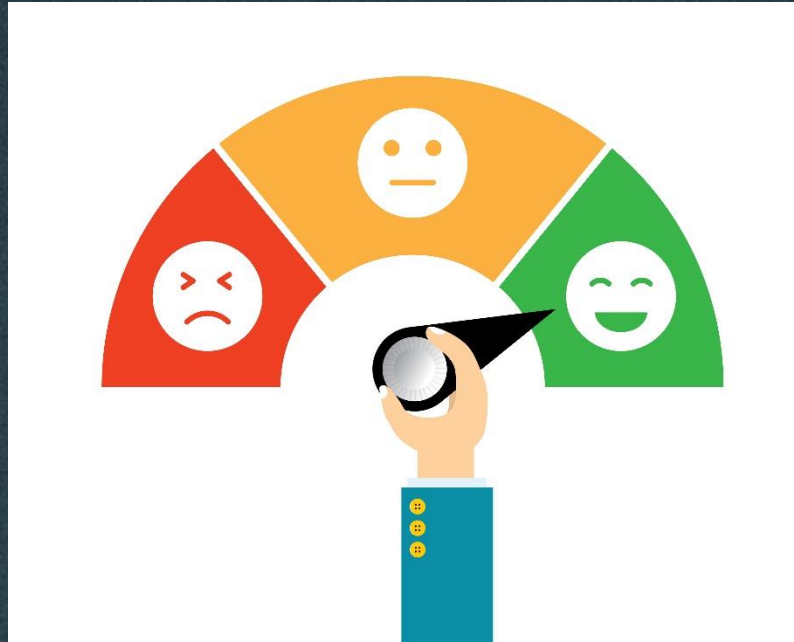


The Agile Manifesto argues that although the concepts on the right have value, those on the left have greater value.





Agile 12 Principles



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



Deliver working software frequently, at intervals of between a few weeks to a few months, with a preference to the shorter timescale.



Working Software is the primary measure of progress

Welcome changing requirements, even late in development.
Agile processes harness change for the customer's competitive
advantage





Continuous attention to technical excellence and good design enhances agility



Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.



Simplicity—the art of maximizing the amount of work not done—is essential.



Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done



The best architectures, requirements, and designs emerge from self-organizing teams



Business people and developers must work together daily throughout the project



The most efficient and effective method of conveying information to and within a development team is face-to-face conversation



At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly



Whole-Team Approach

- Involve everyone necessary to ensure success
- Small team (from 3 to 9)
- Team is Co-located
- Quality is everyone's responsibility



- **Tester's role:**

- support and collaborate with business representatives to help them create suitable acceptance tests
- work with developers to agree on the testing strategy, and decide on test automation approaches
- transfer and extend testing knowledge to other team members and influence the development of the product



- **Power of Three:**

The concept of involving testers, developers, and business representatives in all feature discussions





Early & Frequent Feedback



- In sequential models, early & frequent feedback is not applied
- the customer does not see the product early
- It's too late to change
- The result is unhappy customer and badly managed team



Use **early & frequent feedback** to:

- Focus on the features with the highest business value and deliver them first
- Manage the team better through transparency
- Discover quality problems early



Extreme Programming



Extreme programming

XP embraces five values to guide development:
communication, simplicity, feedback, courage, and respect.

12 Principles of XP

The Planning
Game

Simple
Design

Pair
Programming

40 Hour
Work Week

Small
Releases

Continuous
Testing

Collective
Code
Ownership

On-Site
Customer

System
Metaphor

Refactoring

Continuous
Integration

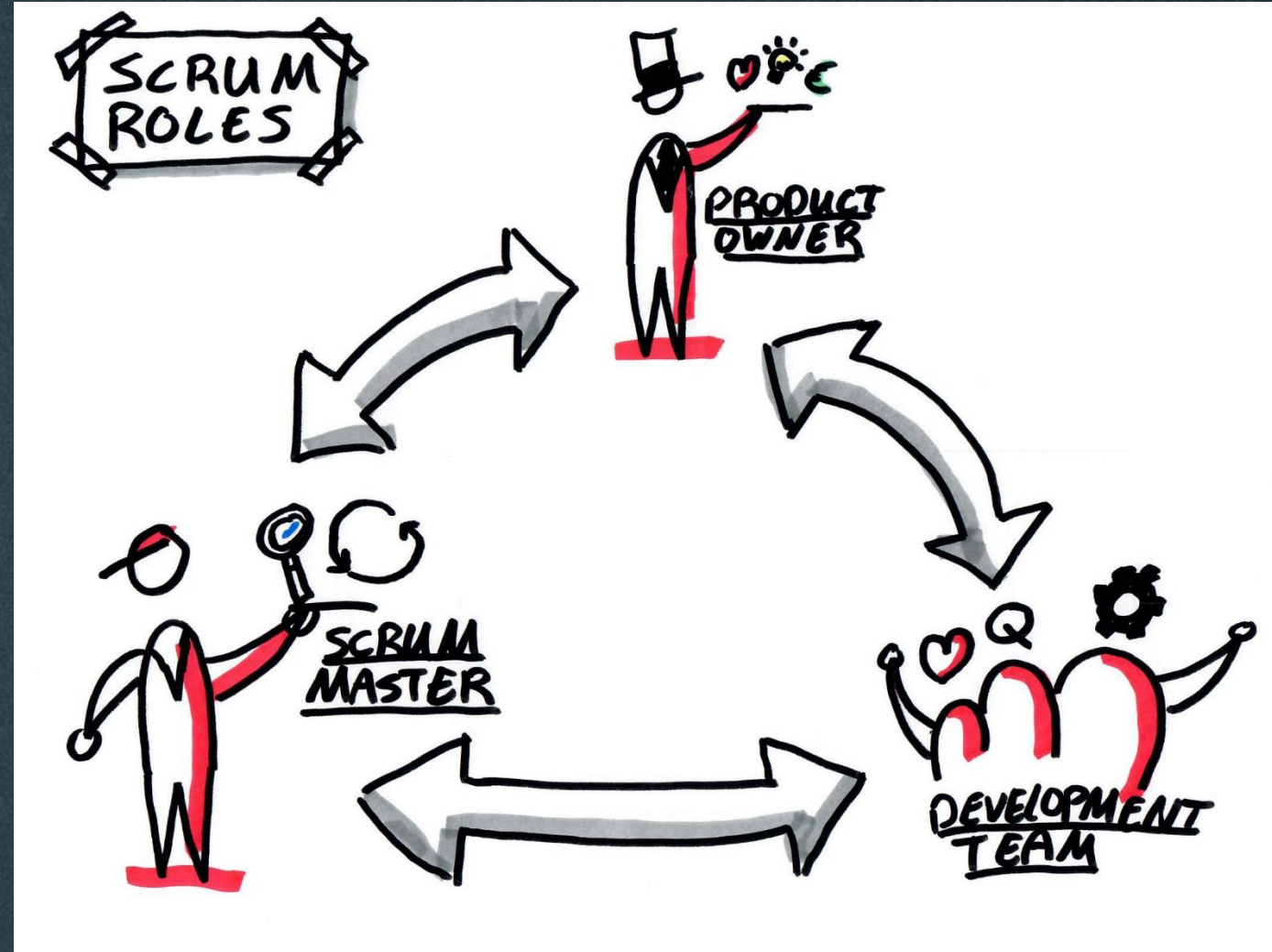
Coding
Standards



Scrum

Scrum Roles:

- 1-Product Owner
- 2-Scrum Master
- 3-Development Team



Product Owner

Product Owner is accountable for maximizing the value of the product resulting from the work of the Scrum Team. How this is done may vary widely across organizations, Scrum Teams, and individuals.

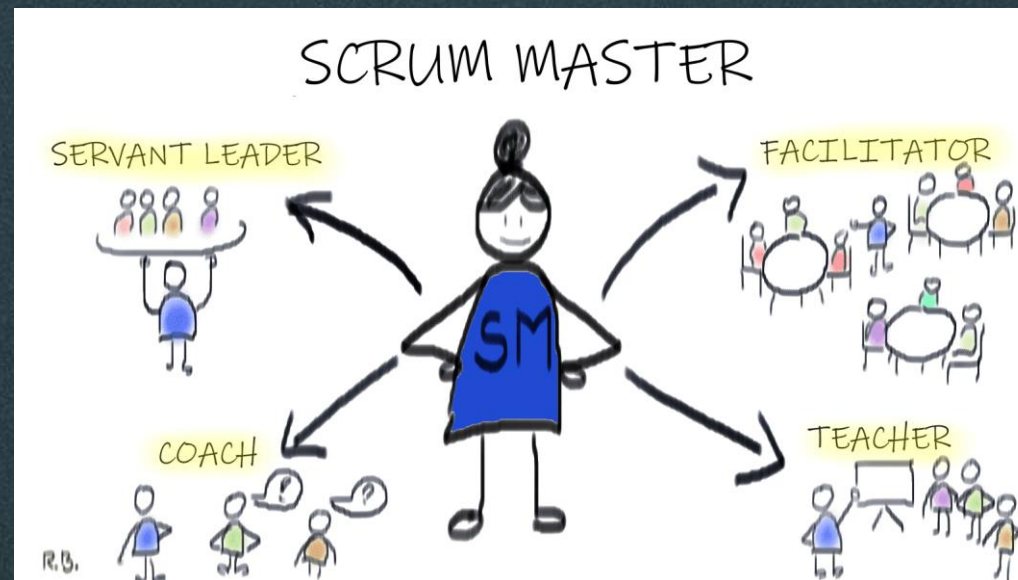
The Product Owner is also accountable for effective Product Backlog management, which includes:

- Developing and explicitly communicating the Product Goal;
- Creating and clearly communicating Product Backlog items;
- Ordering Product Backlog items; and,
- Ensuring that the Product Backlog is transparent, visible and understood.



Scrum Master

- Scrum Master is accountable for establishing Scrum as defined in the Scrum Guide.
- They do this by helping everyone understand Scrum theory and practice, both within the Scrum Team and the organization.
- The Scrum Master is accountable for the Scrum Team's effectiveness. They do this by enabling the Scrum Team to improve its practices, within the Scrum framework.
- Scrum Masters are true leaders who serve the Scrum Team and the larger organization.



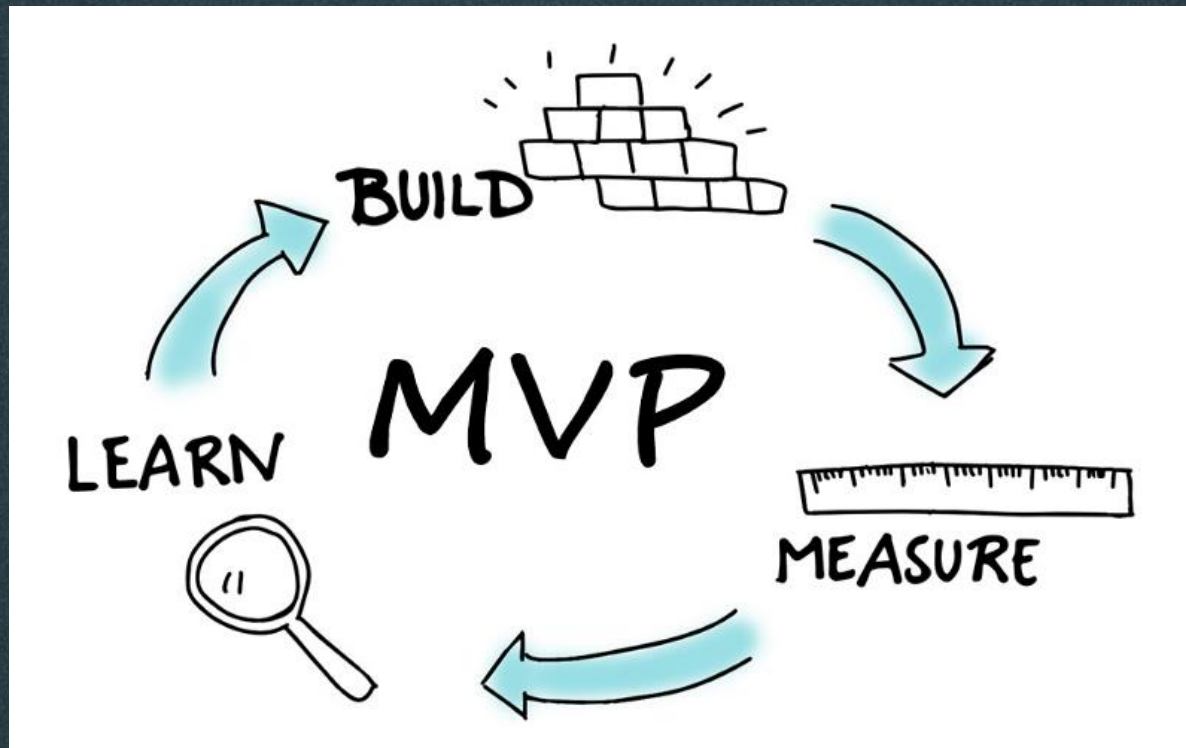
Sprint:

Scrum divides a project into iterations (called sprints) of fixed length (usually two to four weeks).



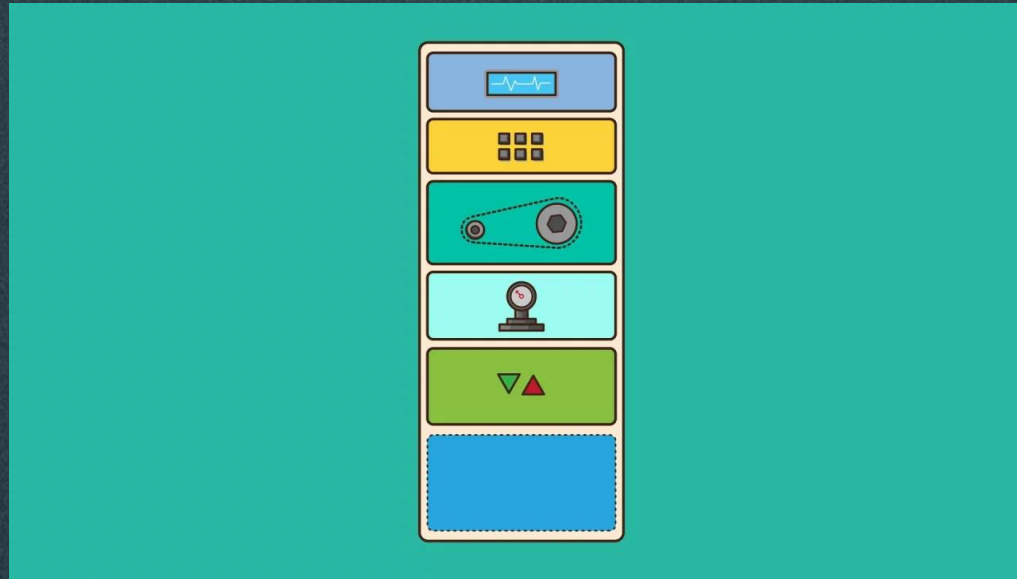
Product Increment:

Each sprint results in a potentially releasable/shippable product (called an increment).



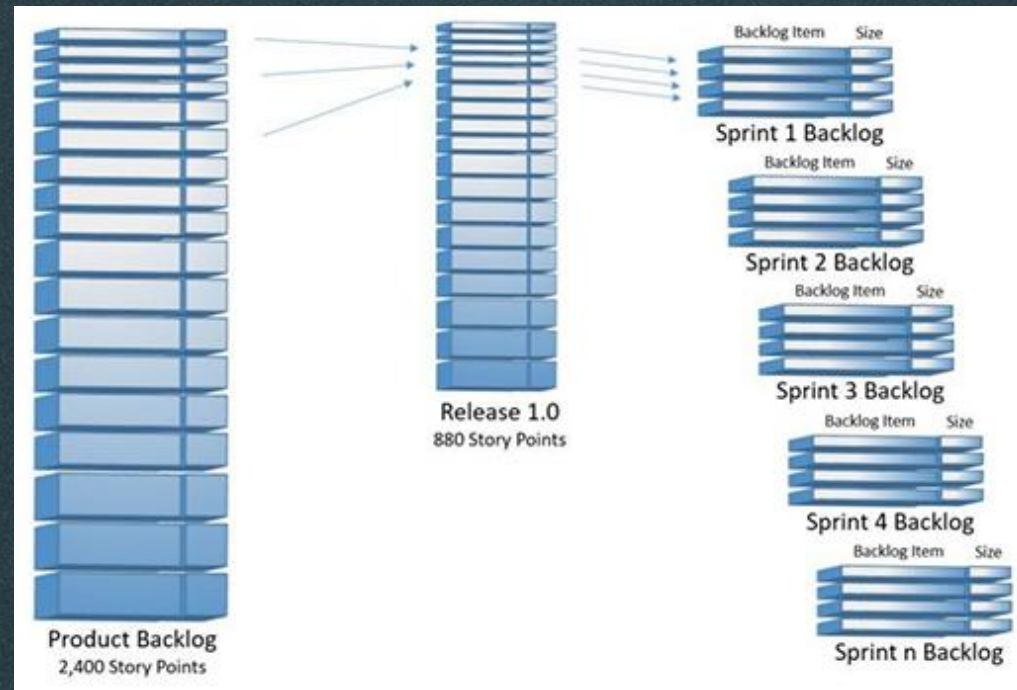
Product Backlog:

The product owner manages a prioritized list of planned product items. The product backlog evolves from sprint to sprint (backlog refinement).



Sprint Backlog:

At the start of each sprint, the Scrum team selects a set of highest priority items from the product backlog. Since the Scrum team, not the product owner, selects the items to be realized within the sprint, the selection is referred to as being on the pull principle rather than the push principle.



Definition of Done:

To make sure that there is a potentially releasable product at each sprint's end, the Scrum team discusses and defines appropriate criteria for sprint completion. The discussion deepens the team's understanding of the backlog items and the product requirements.



Definition of Done

Timeboxing:

Only those tasks that the team expects to finish within the sprint are part of the sprint backlog. If the development team cannot finish a task, it is moved back into the product backlog.



Transparency:

The development team reports and updates sprint status on a daily basis at a meeting called the daily scrum. This makes the progress of the current sprint visible to everyone.





Kanban



Kanban is a management approach that is sometimes used in Agile projects. The general objective is to visualize and optimize the flow of work within a value-added chain.

Kanban Board:

Each column shows a station, which is a set of related activities. The items are symbolized by tickets moving from left to right across the board through the stations.



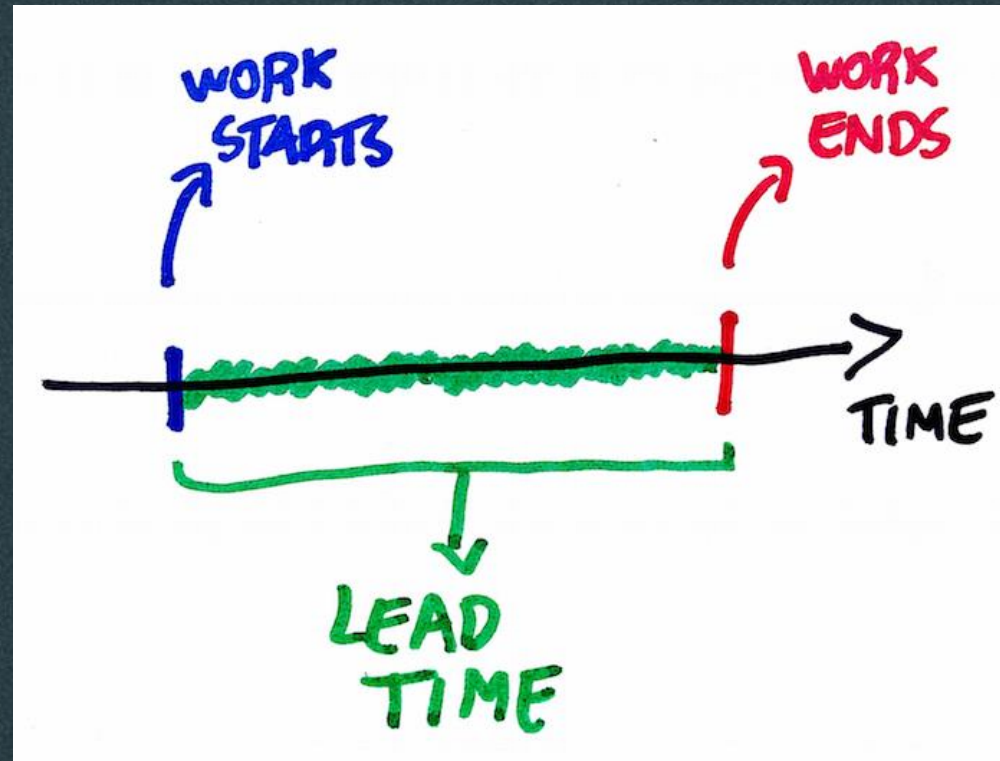
Work-In-Progress Limit:

Each column shows a station, which is a set of related activities. The items are symbolized by tickets moving from left to right across the board through the stations.

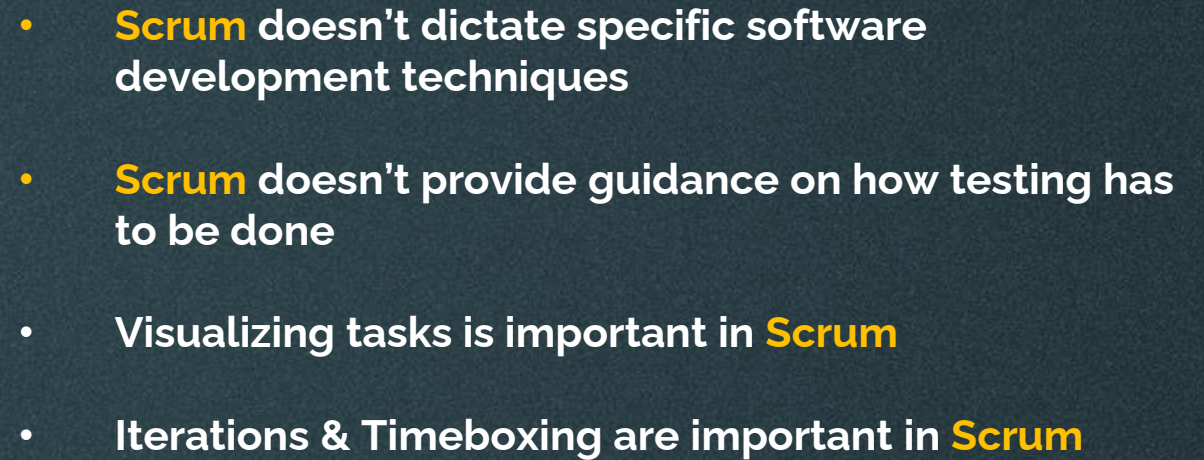
Backlog	Analyze4/5		Develop6/5		Test4/5		Done
<div>+ New item</div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div>	Doing	Done	Doing	Done	Doing	Done	
	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>
	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>
	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div>













Lead Time:

Kanban is used to optimize the continuous flow of tasks by minimizing the (average) lead time for the complete value stream



-



- | Analyze 4/5 | | Develop 6/5 | | Test 4/5 | |
|--|--|--|--|--|--|
| Doing | Done | Doing | Done | Doing | Done |
| <div></div> | <div></div> | <div></div> | <div></div> | <div></div> | <div></div> |
| <div></div> | <div></div> | <div></div> | <div></div> | <div></div> | <div></div> |



User Story

User Stories are written to capture requirements from the perspectives of developers, testers, and business representatives. The user stories must address both functional and non-functional characteristics.



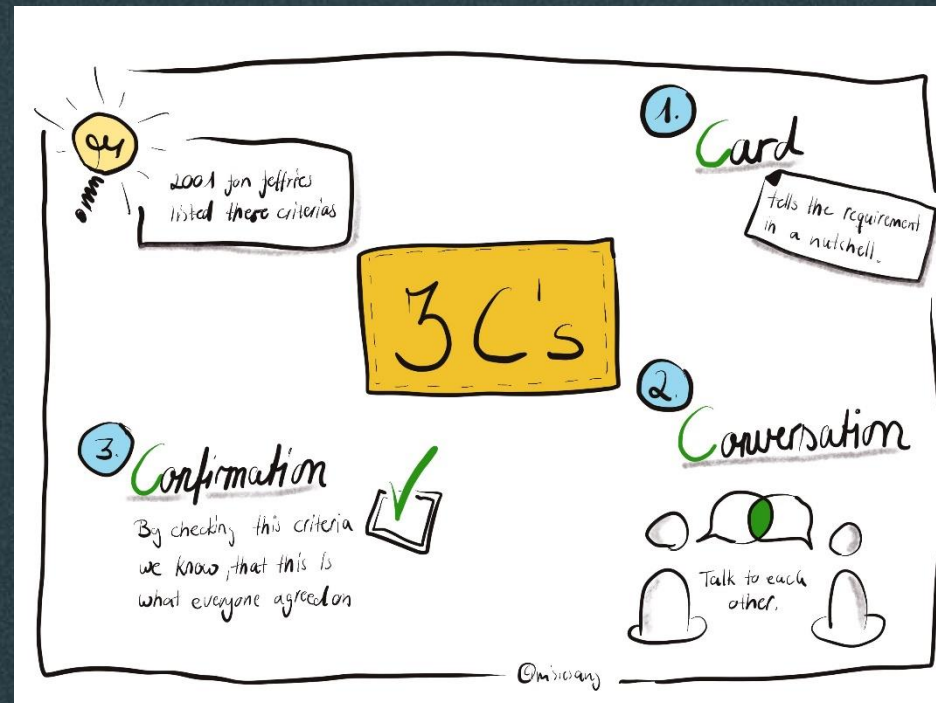
The **Tester's** unique perspective will improve the user story by identifying missing details or non-functional requirements.

The **Tester** can contribute by asking business representatives open-ended questions about the user story, proposing ways to test the user story, and confirming the acceptance criteria.



3C Concept: A user story is a conjunction of three elements:

- 1-Card:** The physical media describing a user story
- 2-Conversation:** Explains how the software will be used
- 3-Confirmation:** The acceptance criteria are used to confirm the user story



Writing a user story

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

User Story

Title:	Priority:	Estimate:
User Story: As a [description of user], I want [functionality] so that [benefit].		
Acceptance Criteria: Given [how things begin] When [action taken] Then [outcome of taking action]		

Title:	Customer Inter Account Transfer
Value Statement:	As a bank customer, I want to transfer funds between my linked accounts, So that I can fund my credit card.
Acceptance Criteria:	<p><u>Acceptance Criterion 1:</u> Given that the account is has sufficient funds When the customer requests an inter account transfer Then ensure the source account is debited AND the target account is credited.</p> <p><u>Acceptance Criterion 2:</u> Given that the account is overdrawn, When the customer requests an inter account transfer Then ensure the rejection message is displayed And ensure the money is not transferred.</p>
Definition of Done:	<ul style="list-style-type: none"> • Unit Tests Passed • Acceptance Criteria Met • Code Reviewed • Functional Tests Passed • Non-Functional Requirements Met • Product Owner Accepts User Story
Owner:	MR I Owner
Iteration:	Unscheduled
Estimate:	5 Points



INVEST Technique

- INVEST Technique:

helps to assess the quality of a user story, the user story should be:

1-**Independent**: of all other user stories (Stories can be worked on in any order)

2-**Negotiable**: A story is an invitation to a conversation

3-**Valuable**: If a story doesn't have a value, it should not be done

4-**Estimatable**: Stories should provide enough information so they can be estimated

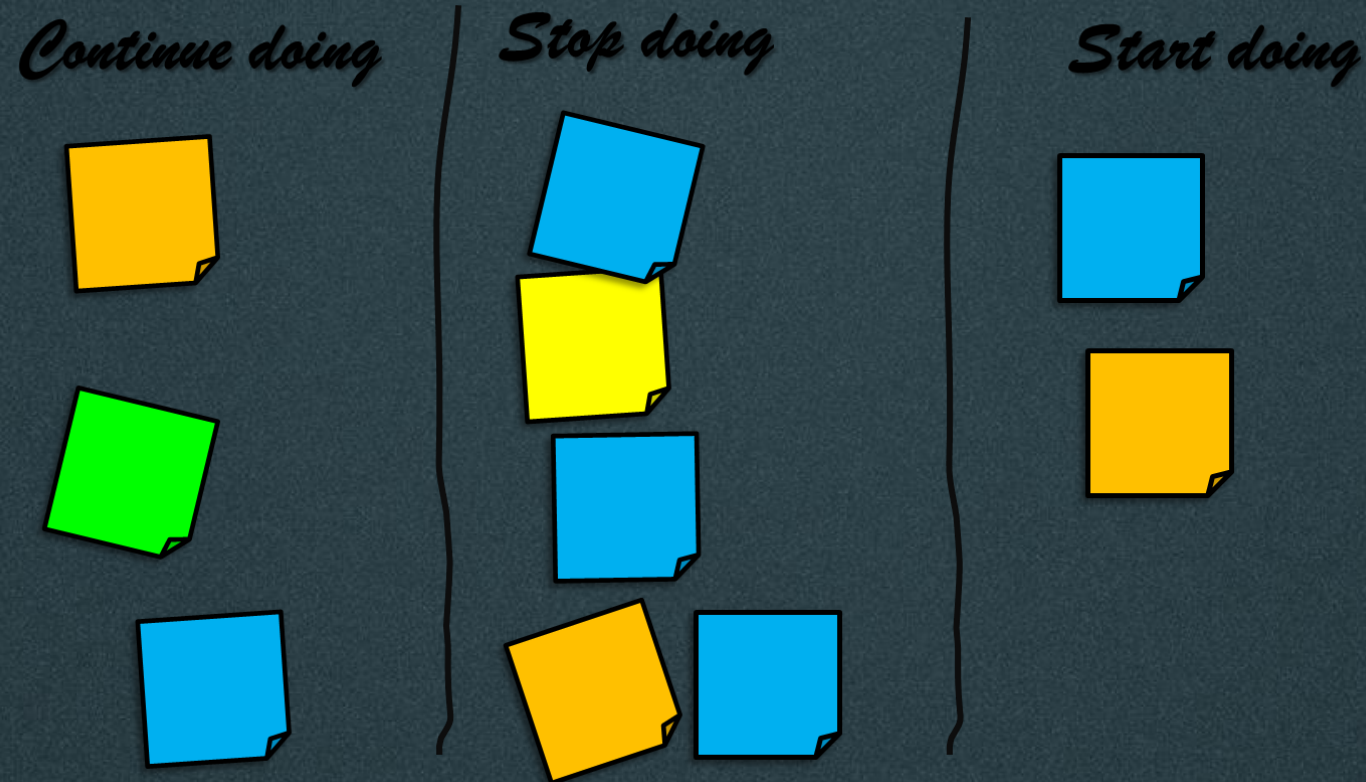
5-**Small**: So as to fit within an iteration

6-**Testable**: Even if the team doesn't have testers



Retrospectives

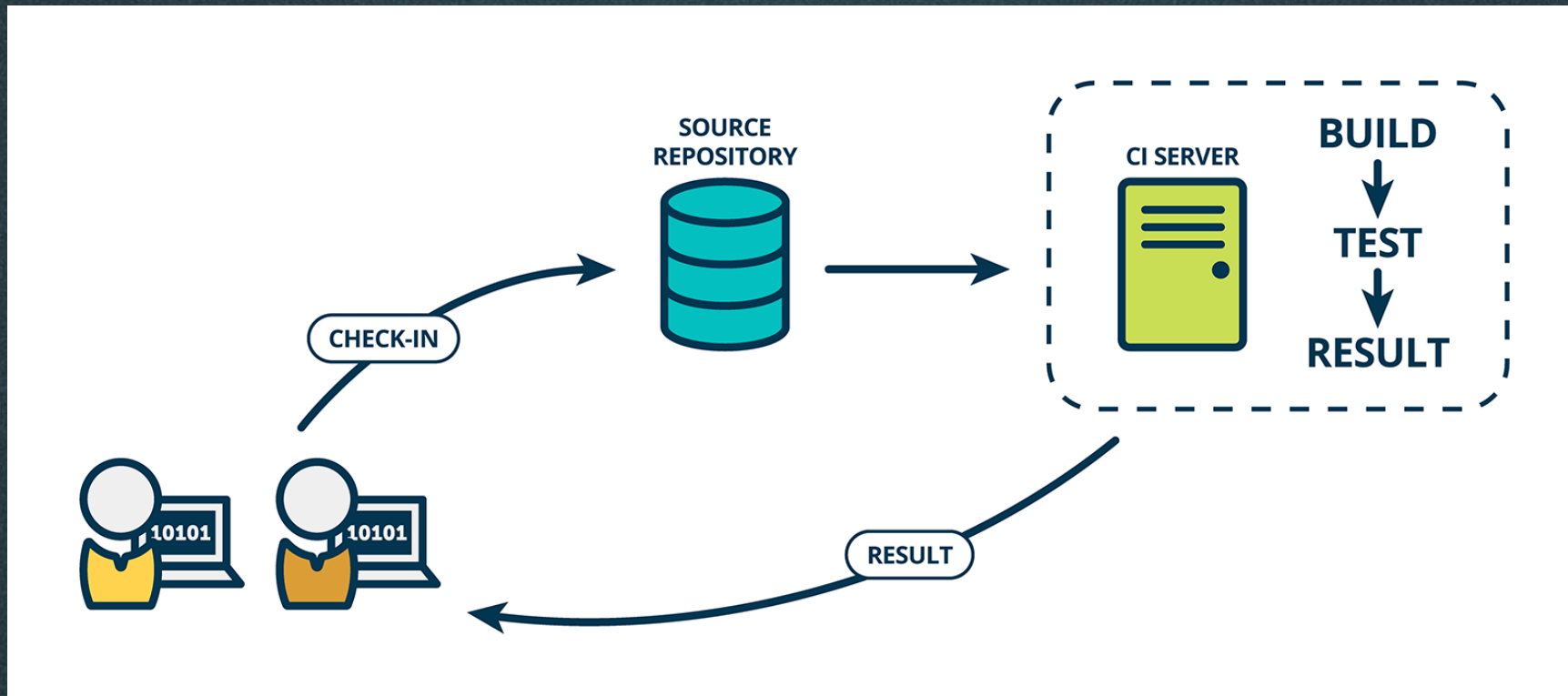
Retrospective is a meeting held at the end of each iteration to discuss what was successful, what could be improved, and how to incorporate the improvements in future iterations.





Continuous Integration

Continuous Integration merges all changes made to the software and integrates all changed components regularly, at least once a day.





Release & Iteration Planning

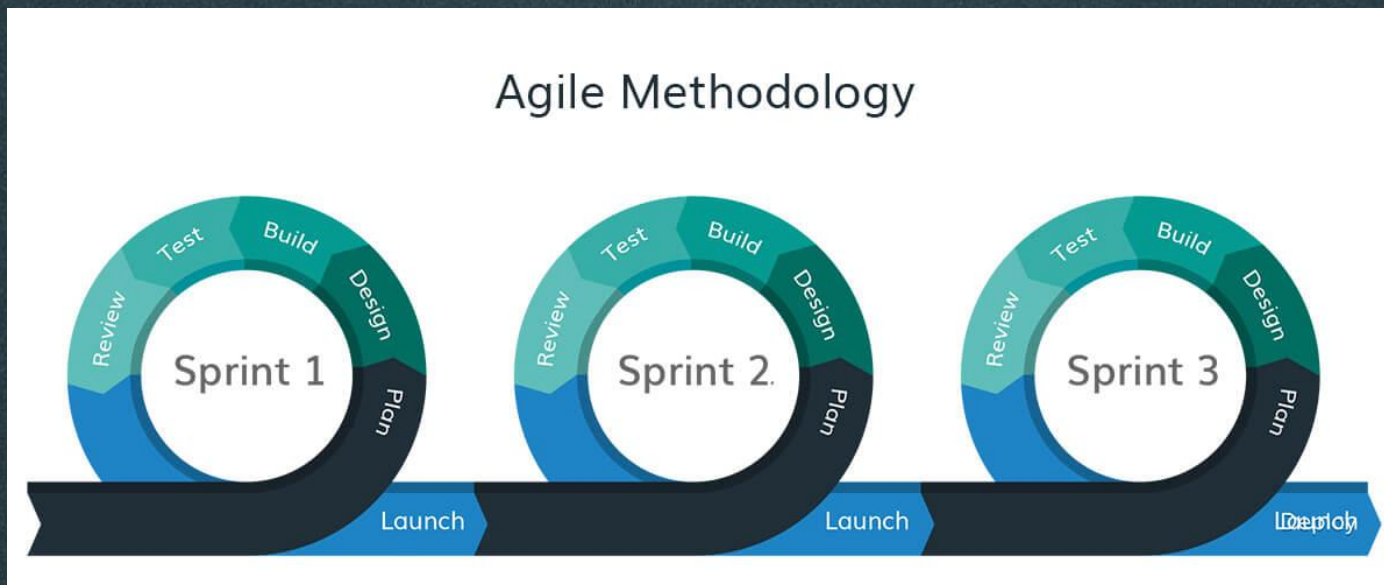
Release Planning looks ahead to the release of a product, often a few months ahead of the start of the project.

Release planning provides the basis for a test approach and test plan.



Iteration Planning looks ahead to the end of a single iteration and is concerned with the iteration backlog.

In **Iteration planning** the team selects user stories from the prioritized release backlog and estimates the work needed for each user story.



JIRA Practical Exercise