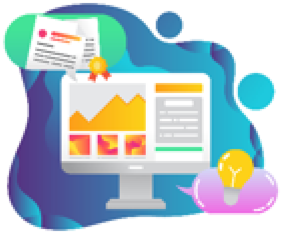
Agile Training 101

Employment
Security
Department
WASHINGTON STATE

Training Agenda

Introduction to Agile 101

- When Agile?
- Agile vs. Waterfall
- Agile in a nutshell
- Why Agile?
- Agile Mindset & Values
- Agile Frameworks



Frameworks: Scrum Overview 101

- High Level view of Scrum to prepare for and contextualize next training

Deep Dive: Scrum Artifacts 201

Artifacts of Scrum

Product Backlog

Sprint Backlog

Product Increment

Acceptance Criteria

Definition of Done/Ready

Deep Dive: Scrum Roles 201

- Scrum roles
- Product Manager / Owner
- Scrum Master
- Development Team
- Scrum Sponsorship

Deep Dive: Scrum Ceremonies 201

- Scrum Ceremonies
- Sprint Planning
- Daily Scrum
- Backlog Refinement
- Sprint Review / Demo
- Sprint Retrospective

Agile in a Nutshell

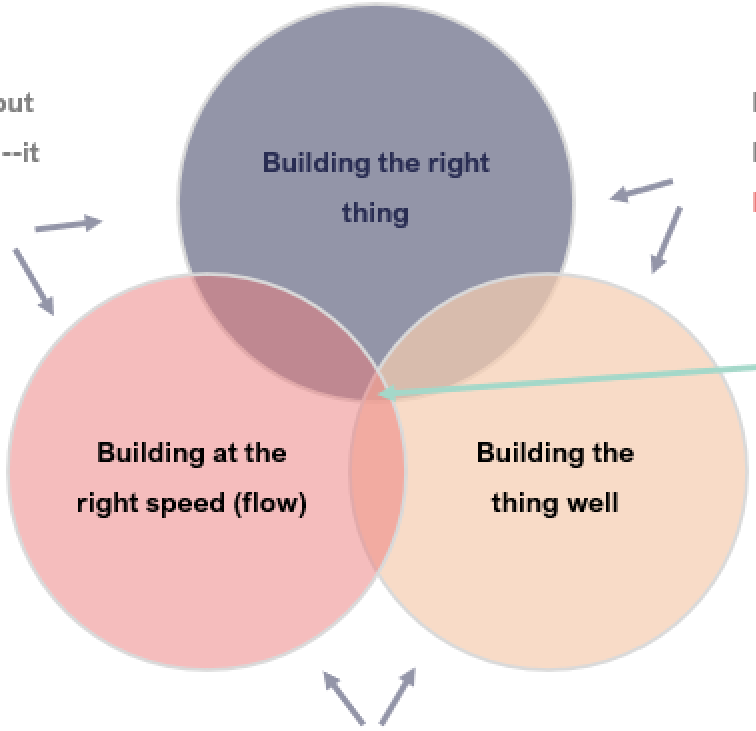
Agile is a way of working that acknowledges:

- *We can’t know everything in advance*
- *…and things we do know—very well may change.*

Agile manages this by prioritizing a specific set of values, and by intervening with socio-technological systems (ceremonies, artifact, and roles) through the use of frameworks that promote learning, which enable organizations to adapt quickly when new information emerges.

How agile works: Balancing competing interests

Building the **right** thing **fast** is great for prototyping, but without quality and a solid architecture--it will **fail over time**



Building the **right** thing **well** is great but without speed it **compromises the budget and timing windows**

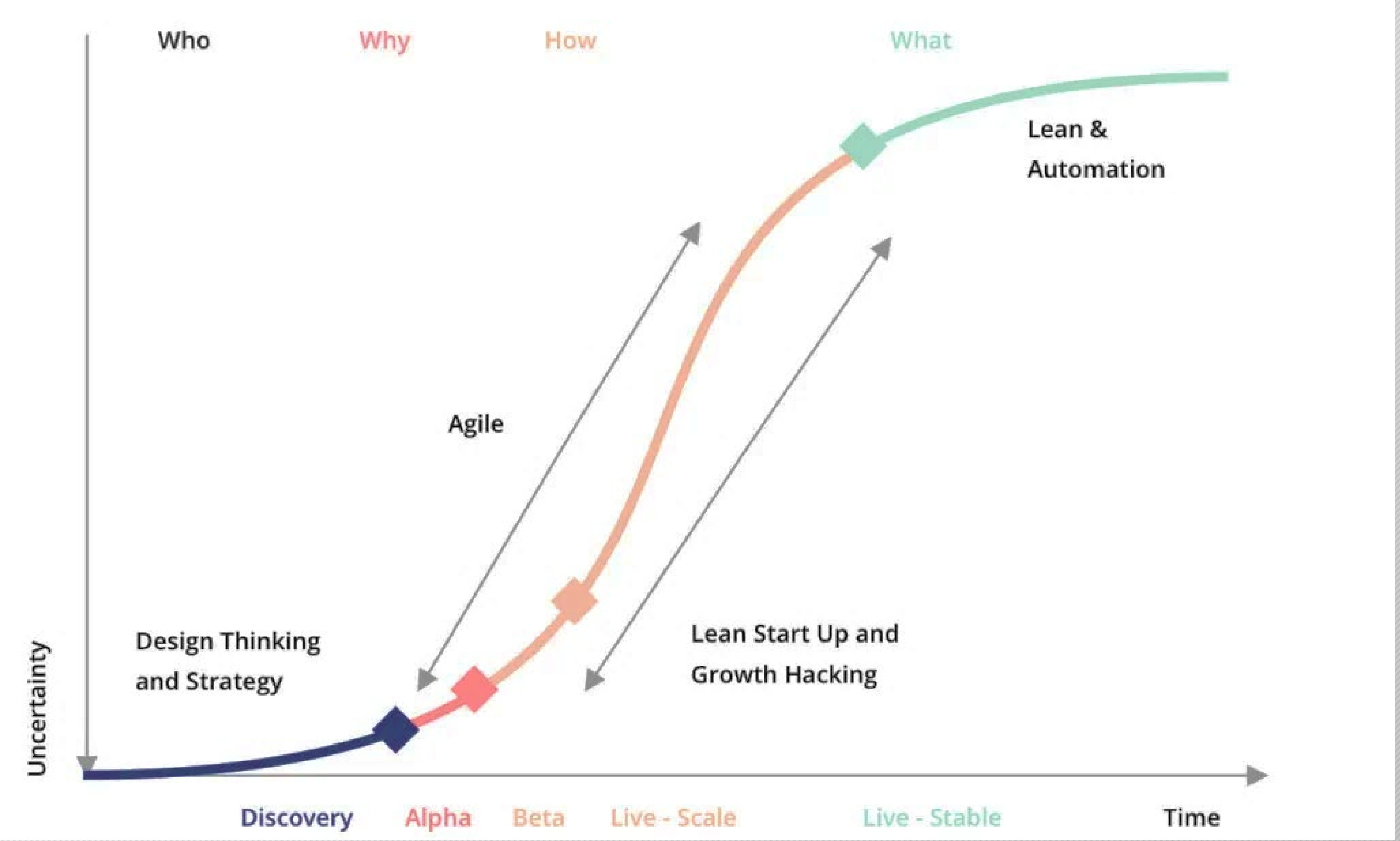
Building the **wrong** thing **fast & well** simply does not work, and it is a **huge waste of time and money**

This is where agile lives!

Agile compiles elements of the three categories to be able to deliver the right thing (supplemented by discovery and design thinking), fast and with quality (elements of lean).

In fact, agile is the bridge between discovery/design thinking and lean phases as seen on the next slide

When to use Agile?



It is all about uncertainty and complexity…

High complexity projects facing uncertainty require a holistic and humble approach to problem solving using design thinking and discovery techniques.

When you know what to do, but you need to find a way how to do it, agile is the answer.

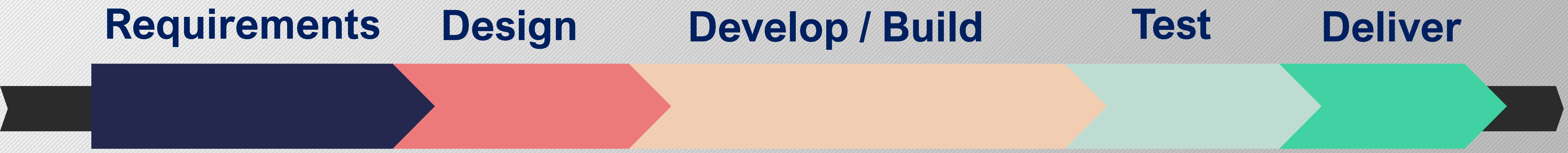
When you know exactly what to do, and how to do it, Lean and Automation are your allies.

Agile vs. Waterfall

To explain agile, it helps to have a frame of reference.

Waterfall has been a standard way of delivering technology projects for many decades. One department delivers one part of the project, while the others wait until it is executed.

Each stage also requires an approval. If the delivery is not approved, the project must come back to re-adjust, re-do work potentially back to the beginning of the phase (or even earlier phases). Other phases keep on hold. Delivery is not usually tested until integrated and delivered to the customer.



Requirements

Design

Develop / Build

Test

Deliver

Agile vs. Waterfall

Waterfall uses the theory that what you want at the beginning is what you get at the end so there is little, if any room, for significant changes in direction, leading to a death march. Waterfall failure points:

Leads to a One-Way Street

- This model is just like the one-way street. Once phase X is completed, and next phase Y has started then there is no way to going back on the previous phase.

Does not factor in Overlap

- The waterfall model lacks an overlap among phase. The waterfall model recommends that new phase can start only after completion of the previous phase. **But in real projects**, this can’t be maintained. To increase the efficiency and reduce the cost, phases overlap, creating a theory practice gap.

Low Interaction

- The waterfall model lacks interaction among phase. Users have little interaction with project phases. Feedback is not taken during development. After a development process starts, changes can not accommodate easily.

Big bang delivery

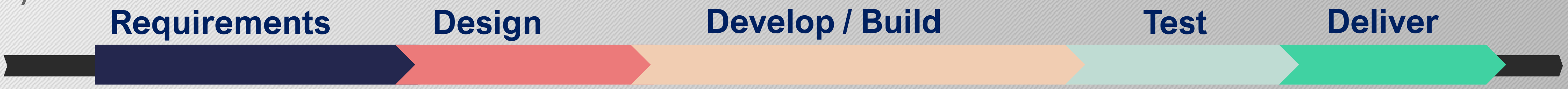
- Does not support delivery of system in pieces. After a development process starts, changes cannot be accommodated easily.

Feedback Loops?

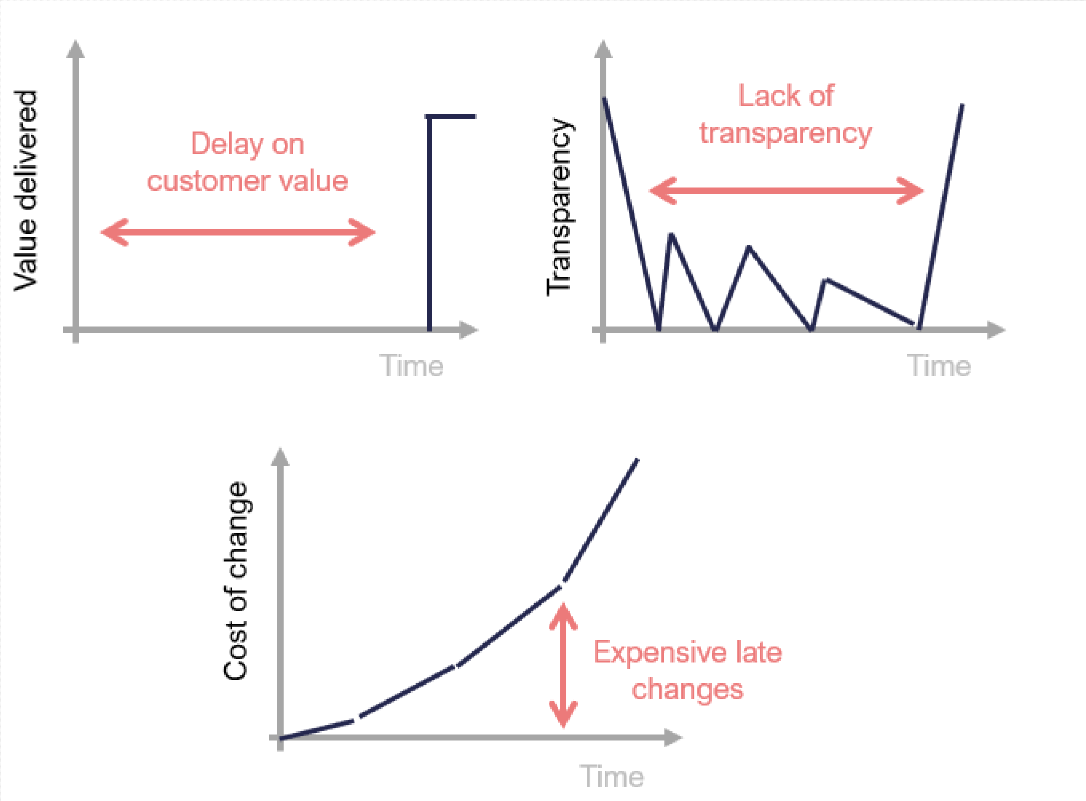
- The waterfall model has no feedback loops. Waterfall assumes that no error is ever committed by during any phase. Hence, it does not incorporate any mechanism for error correction.

Inflexible

- Difficult to accommodate change requests. Waterfall assumes that all the customer requirements **can be** completely and correctly defined at the beginning of the project, but customers’ requirements keep on changing with time. After the requirements specification phase is completed difficult to accommodate any change requests.

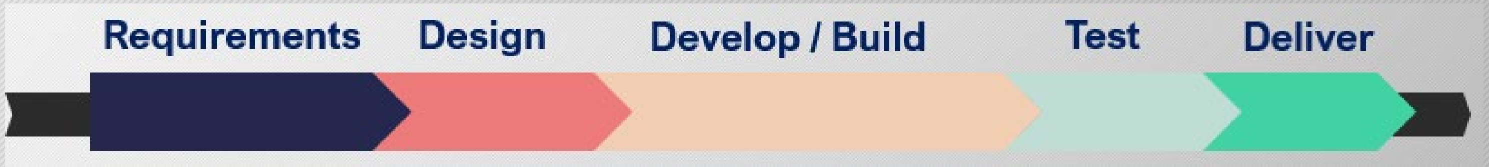


Agile vs. Waterfall

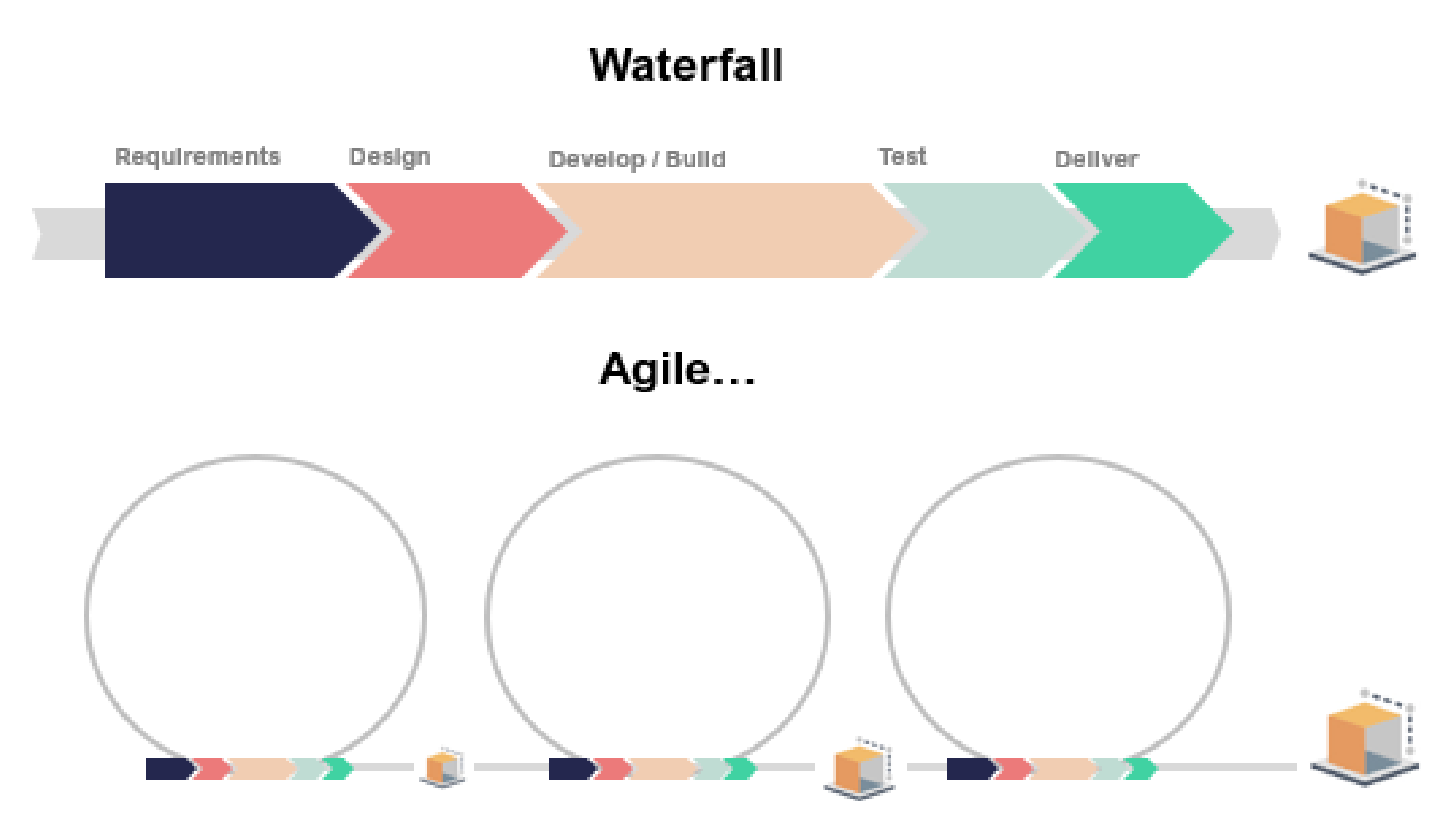


In uncertain environments, the waterfall approach has several downsides, especially when requirements change. This leads to several consequences:

- Delay on customer value
- Lack of transparency to customer
- High cost of change and modifications after delivery



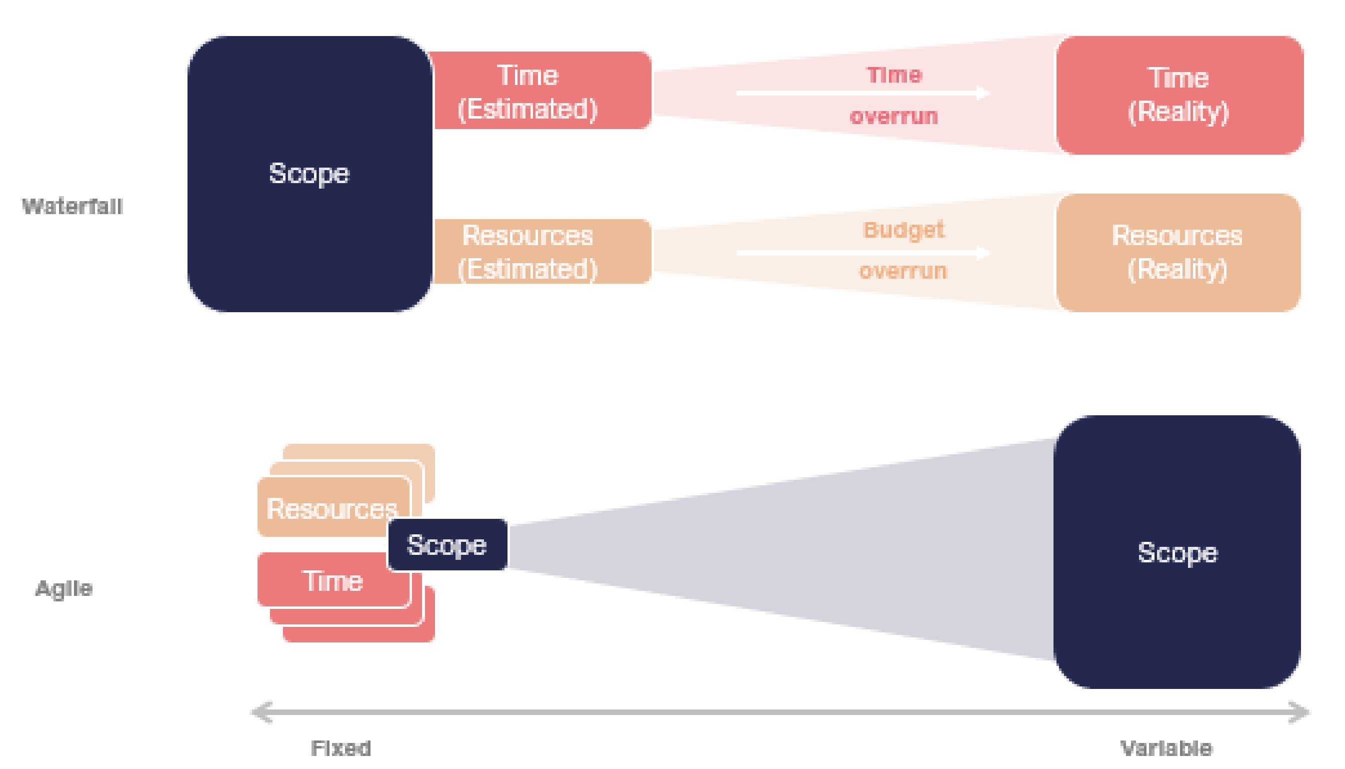
Agile vs. Waterfall



While waterfall focuses on a step-by-step sequence of a full-fledge delivery, agile focuses on delivering all sequence in a short cycle, with few highest value-adding features.

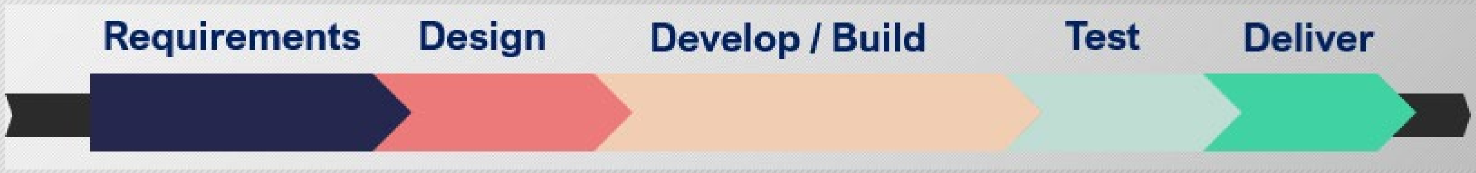
The incremental nature, with frequent deliveries, is one of the key differences when working agile

Agile vs. Waterfall

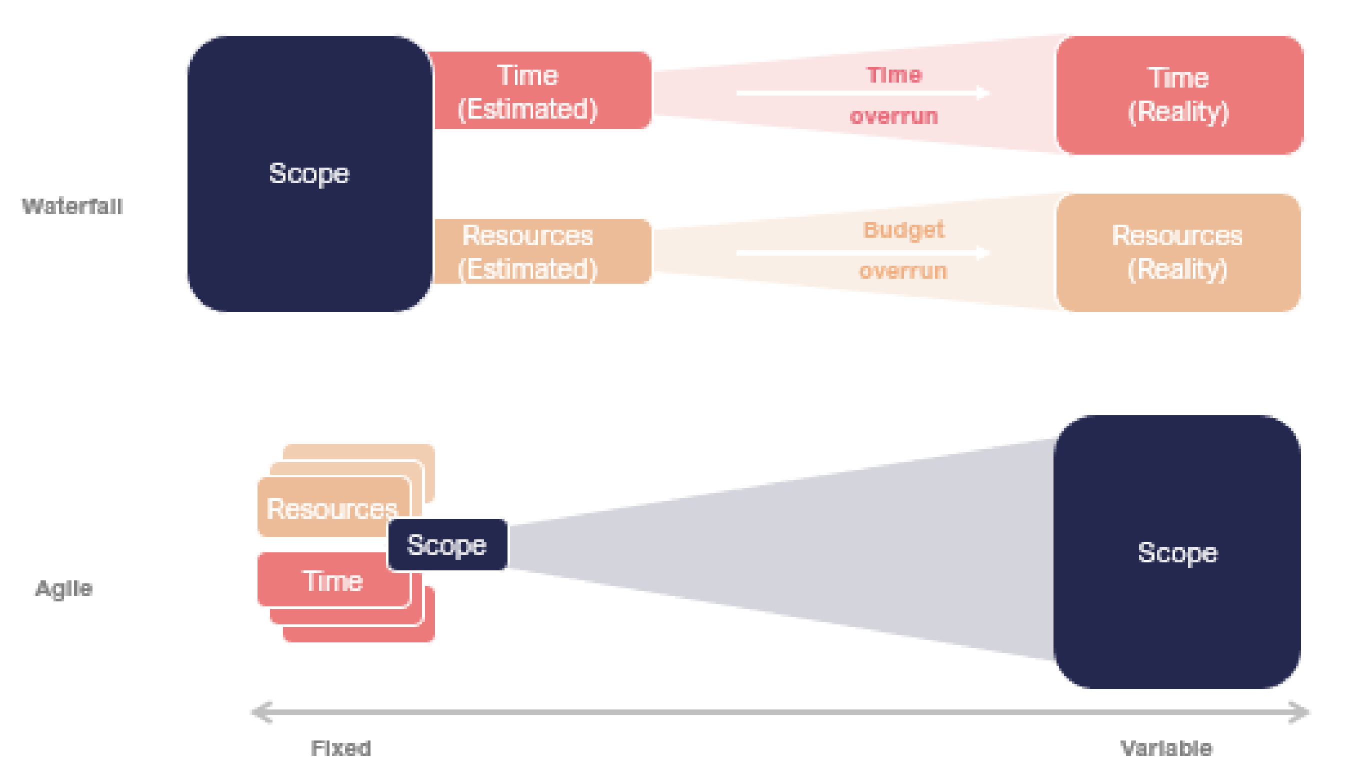


For decades we have heard many projects that went over budget, over time and have failed to meet customer needs.

Agile reduces time delivery by allocating small amount of resources and time to deliver small fixed scopes. Over time, the aggregation of those scopes is a larger delivery with a potentially different scope.

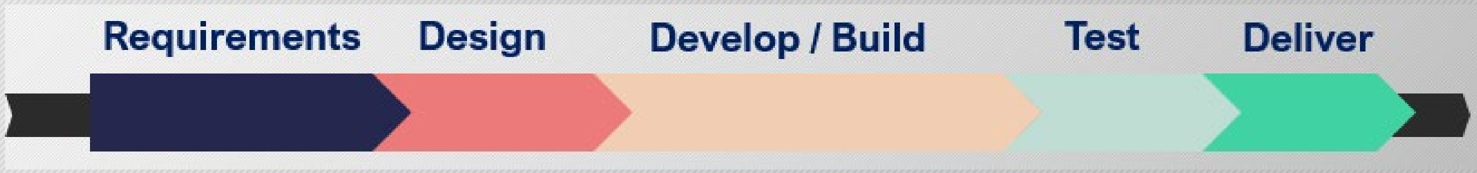


Agile vs. Waterfall



For decades we have heard many projects that went over budget, over time and have failed to meet customer needs.

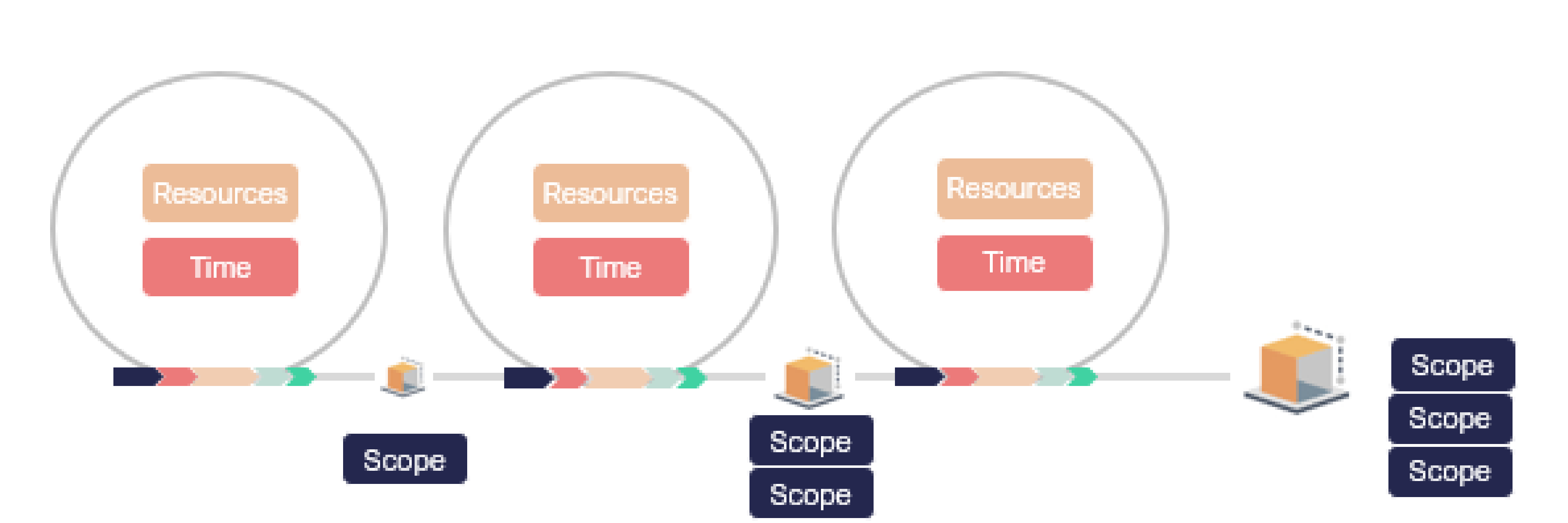
Agile reduces time delivery by allocating small amount of resources and time to deliver small fixed scopes. Over time, the aggregation of those scopes is a larger delivery with a potentially different scope.



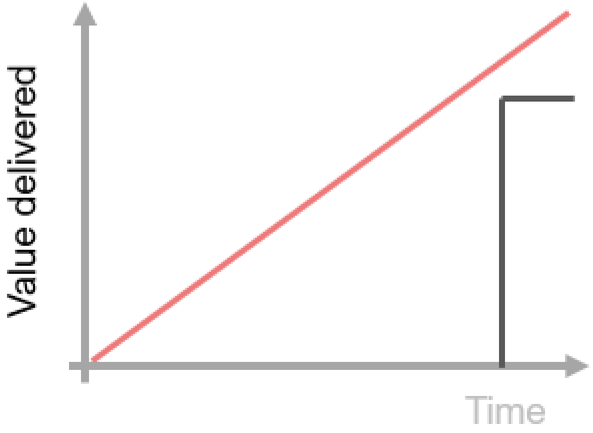
Agile vs. Waterfall

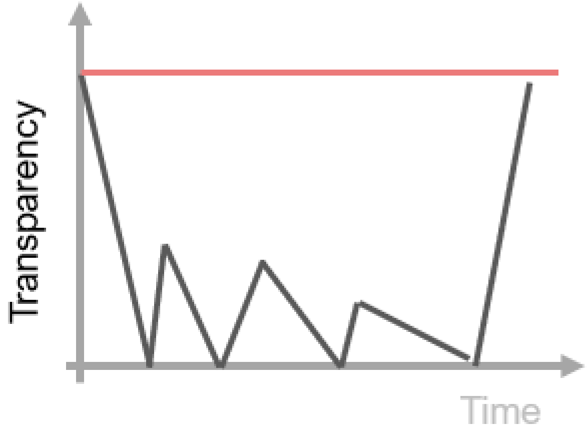
While waterfall focuses on a step-by-step sequence of a full-fledge delivery, agile focuses on delivering all sequence in a short cycle, with a few, high value adding features….

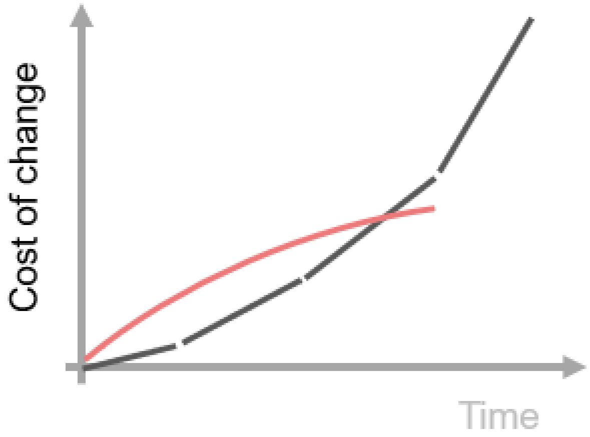
The incremental nature, with frequent deliveries, is one of the key differences when working agile



Agile vs. Waterfall







In uncertain environments, has several upsides compared to waterfall:

Earlier customer delivery of value

Increased transparency

Increased testing frequency

Lower cost of change and modification

Why Agile?

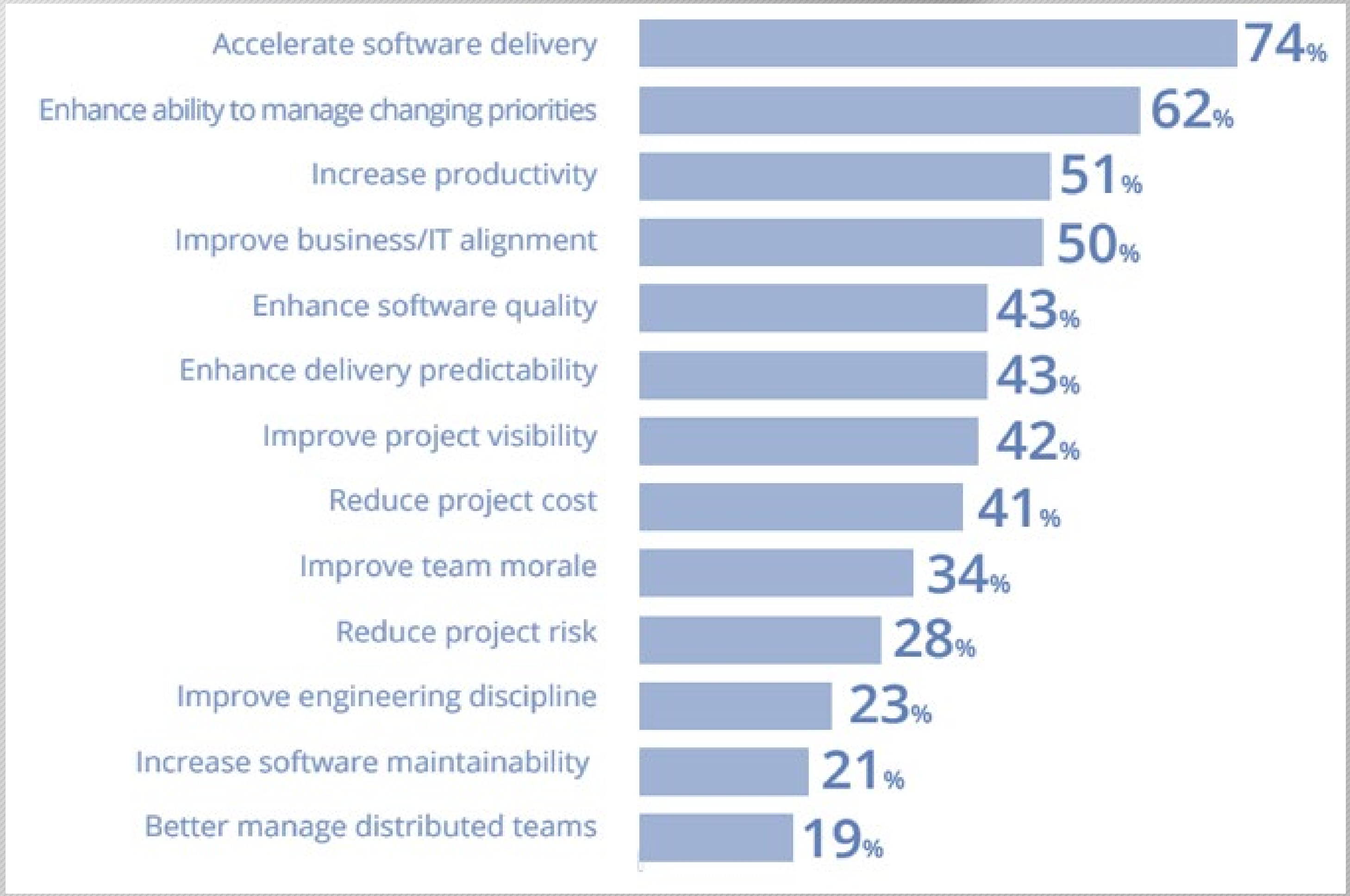
In the face of complexity and uncertainty, agile is just a better way of delivering technology solutions

In my experience it achieves…

- Higher value creation through continuous discovery, prioritization, and validation of work with the people who are using and impact by the solution.
- Improves team collaboration through ceremonies and artifacts
- Lower risk by delivering small increments tested and validated against the users at a regular cadence

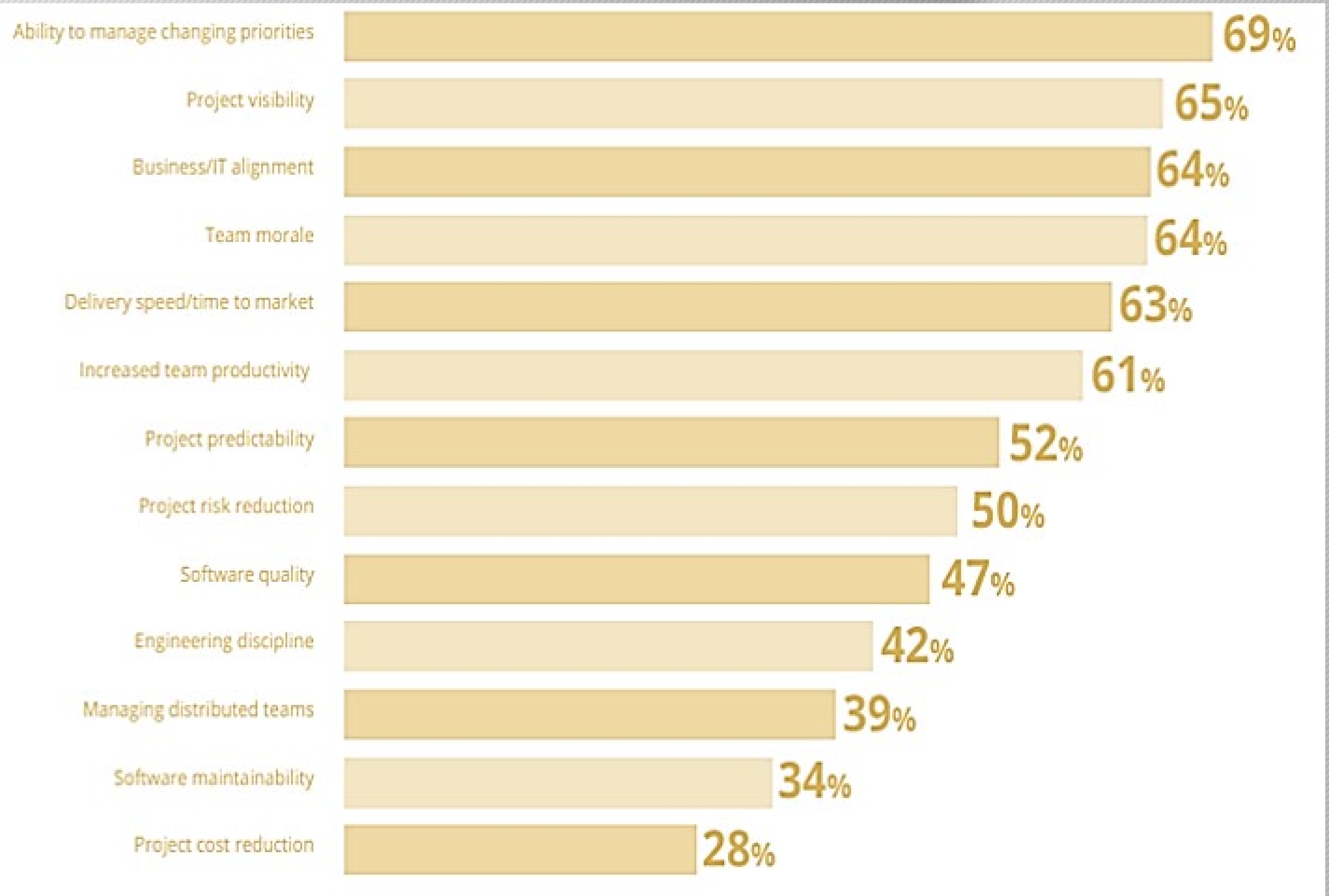
Why Agile?

Organizations state several key motivations or reasons for adopting agile.



Why Agile?

And several concrete benefits realized after adopting agile.

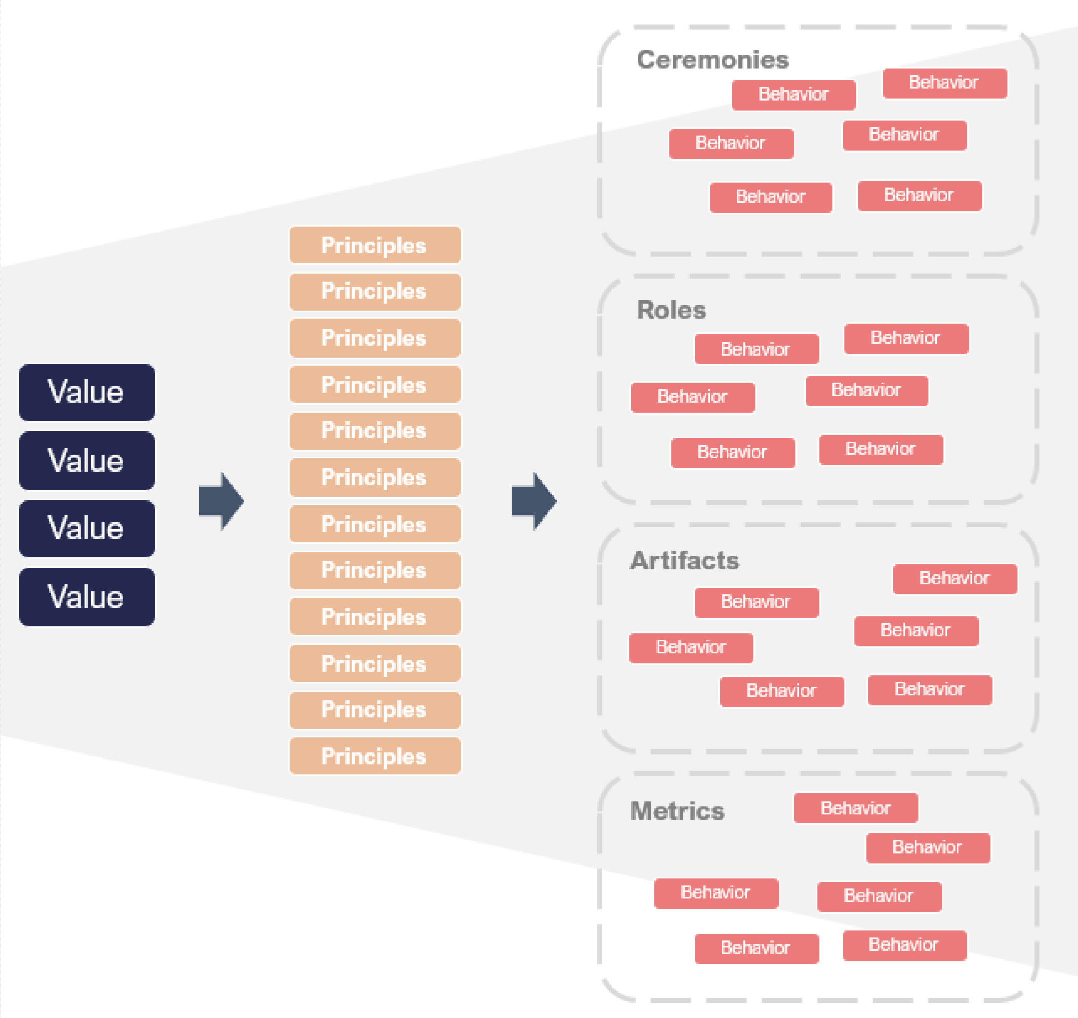


Agile Values

Agile's success is in the values, principles and team dynamics realized through reinforcing behaviors and core beliefs…

…so let's talk about those values and principles.

Agile Pyramid



Agile is a philosophy put in practice through demonstrating behaviors.

To help us grasp what being agile means the agile manifesto states:

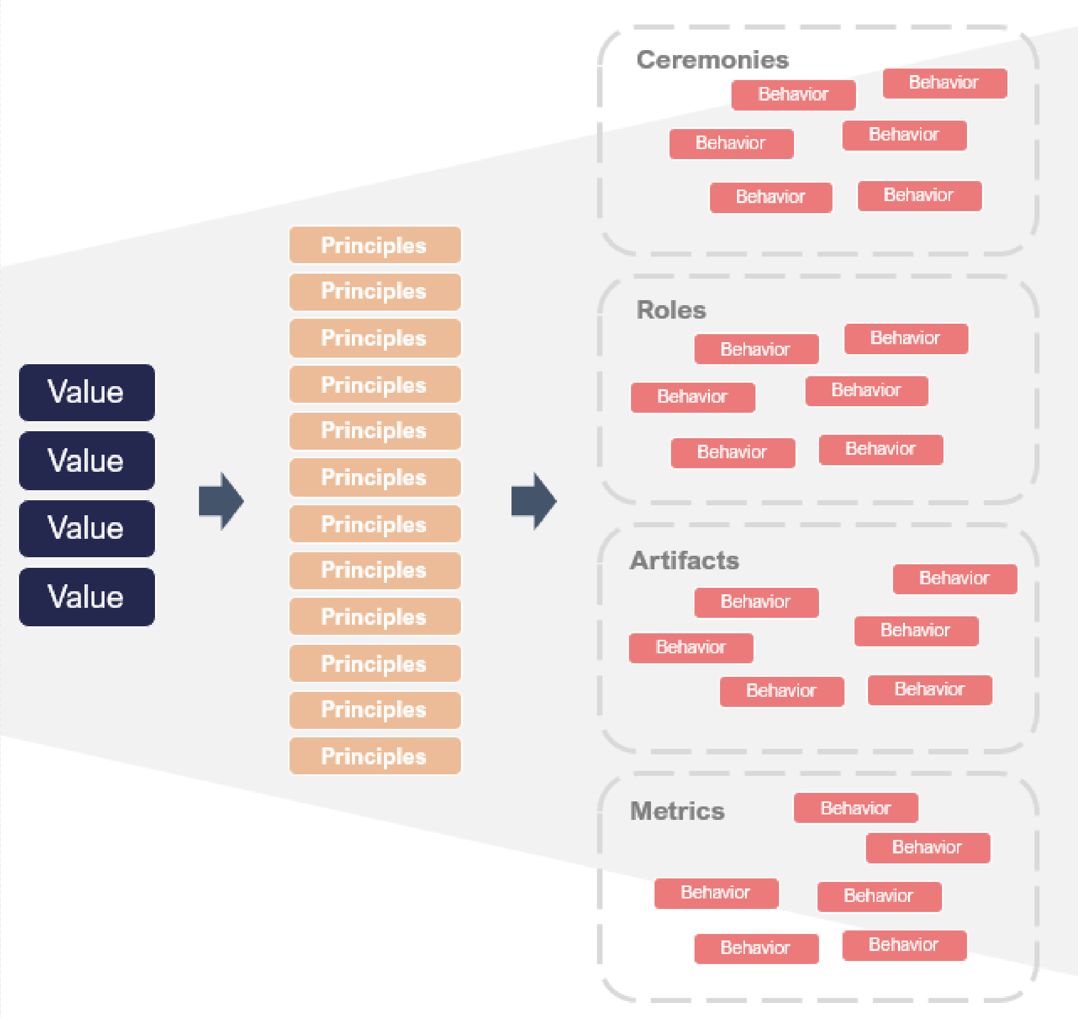
4 values

12 principles.

To help us be agile, several practices or frameworks have been developed (e.g. Scrum).

These frameworks contain certain roles, artifacts, ceremonies, and metrics

Agile Pyramid



Typical outcomes of these types of behaviors are:

- Focusing on delighting the customer
- Building high performing teams
- Inspect and adapt for learning and correction
- Incremental deliveries for short feedback loops
- Nurture a continuous improvement culture

…so what are these values?

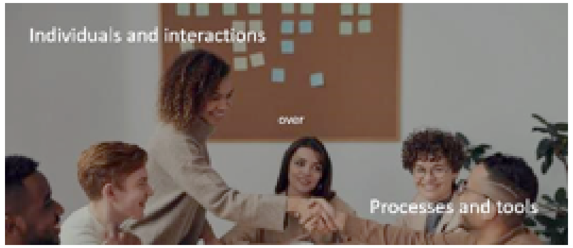
Agile Values: Manifesto

“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:”

Individuals and interactions		Processes and tools
Working software		Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change		Following a plan

…that is, while there **is value** in the items on the right, we value the items on the left **more**.

12 agile principles, grouped by agile values

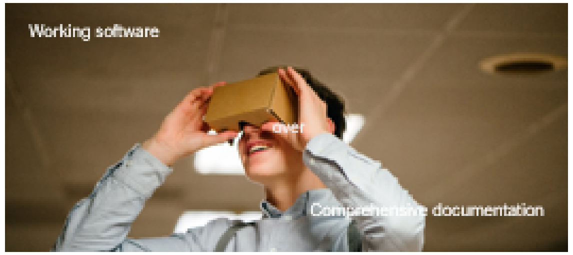


Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

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

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

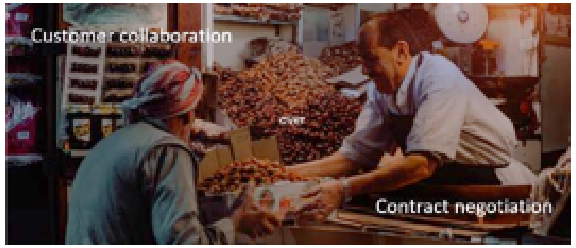
Build projects around motivated highly competent individuals. Give them the environment and support they need, and trust them to get the job done.



Working software is the primary measure of progress.

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

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.



Welcome changing requirements, even late in development. Agile processes harness change as competitive advantage.

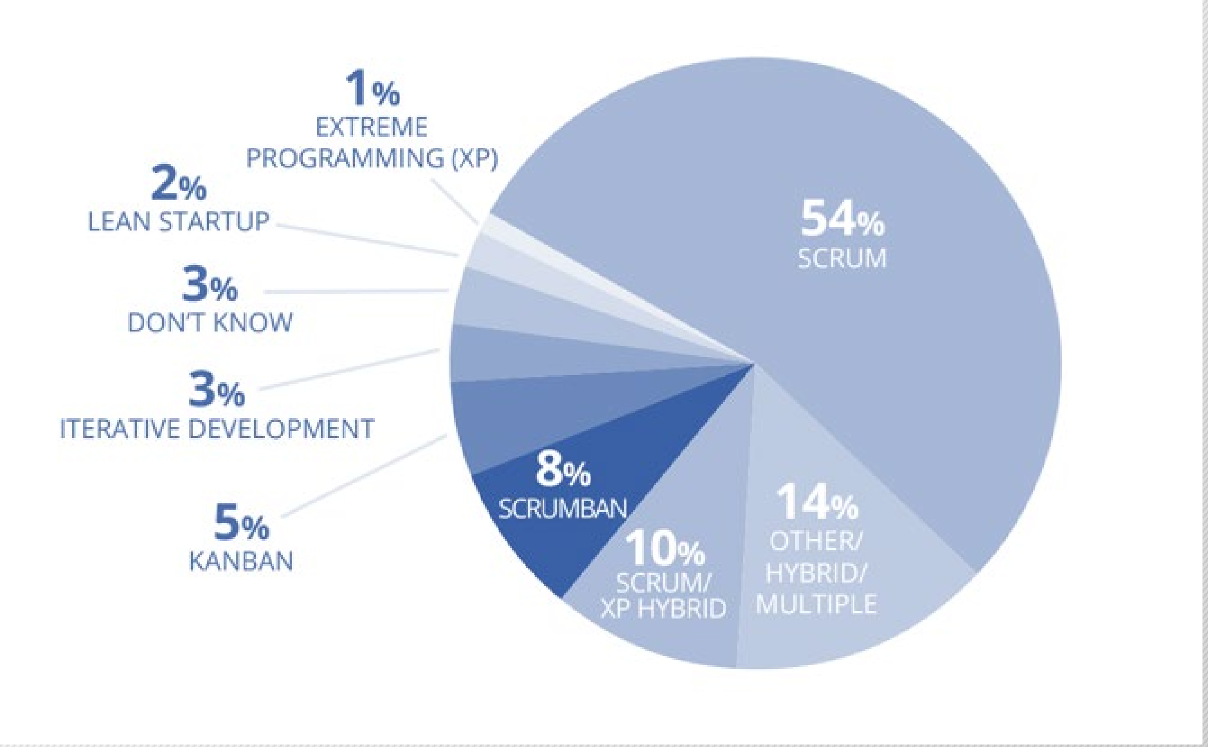
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Continuous attention to technical excellence and good design enhances agility.

Businesspeople and developers must work together daily throughout the project.

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

Agile Frameworks



Agile behaviors are “encoded” into methodologies or practices.

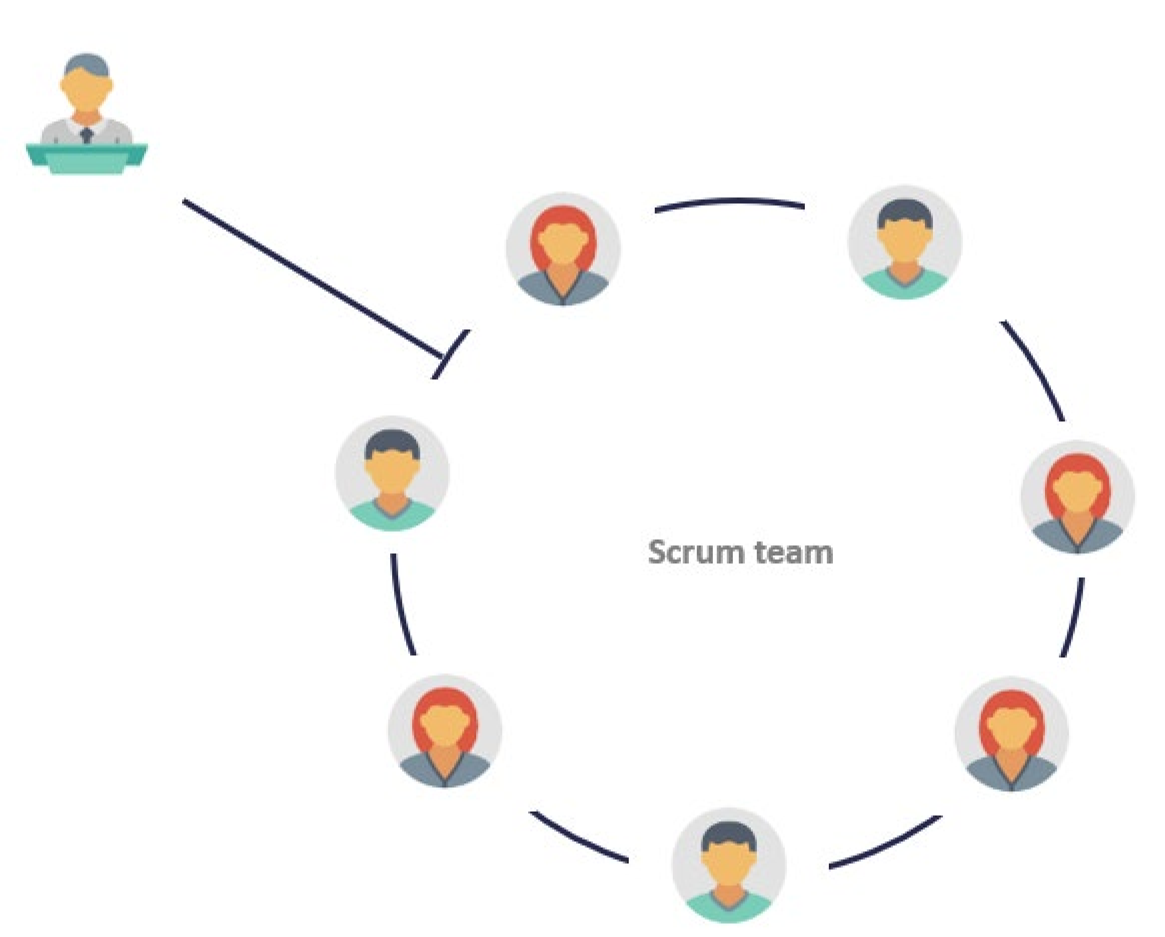
There are several agile frameworks to choose from.

In my humble opinion, Kanban is not an agile framework, as it comes from Lean, and it does not have an iterative nature, which I consider fundamental to agile.

The most widely used practice is Scrum. I find it relatively easy to adopt, although difficult to master.

Let’s explore Scrum a bit.

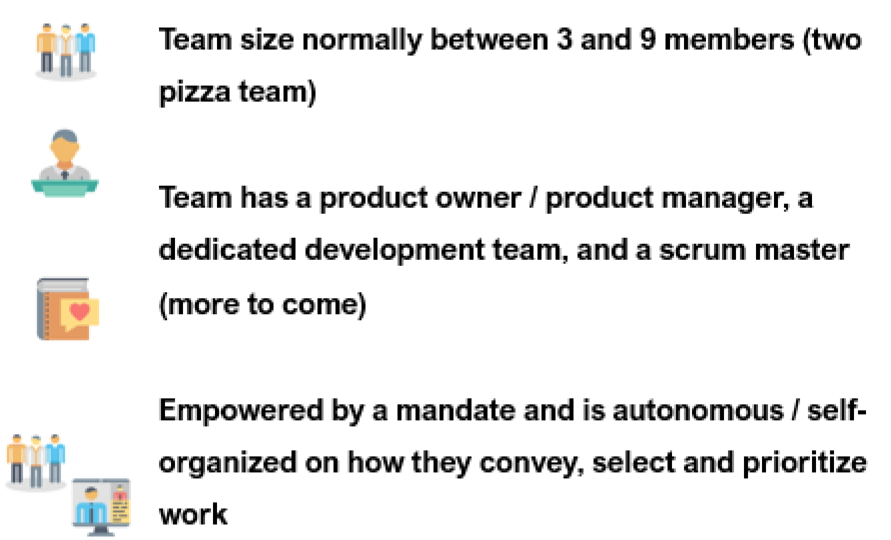
Scrum roles: Scrum Team



The creation of Scrum teams is critical to agile success.

Agile teams are highly interdependent - they plan work, solve problems, make decisions, and review progress in service of a specific project. Team members need one another to get work done.

An agile team is typically set under the following criteria:



Team size normally between 3 and 9 members (two pizza team)

Team has a product owner / product manager, a dedicated development team, and a scrum master (more to come)

Empowered by a mandate and is autonomous / self-organized on how they convey, select and prioritize work

Artifacts

Scrum artifacts are tools that help the product manager and the scrum team manage their work

Product Backlog

The product backlog is created from the product manager / product owner, supported by findings from design thinking and continuous discovery work

The product manager prioritizes the items in the backlog according to the vision and the value proposition to the customer

Sprint Backlog

The sprint backlog is a list of tasks identified by the team to be completed during the sprint.

During the sprint planning meeting, the team selects some number of product backlog items, usually in the form of user stories, and identifies the tasks necessary to complete each user story.

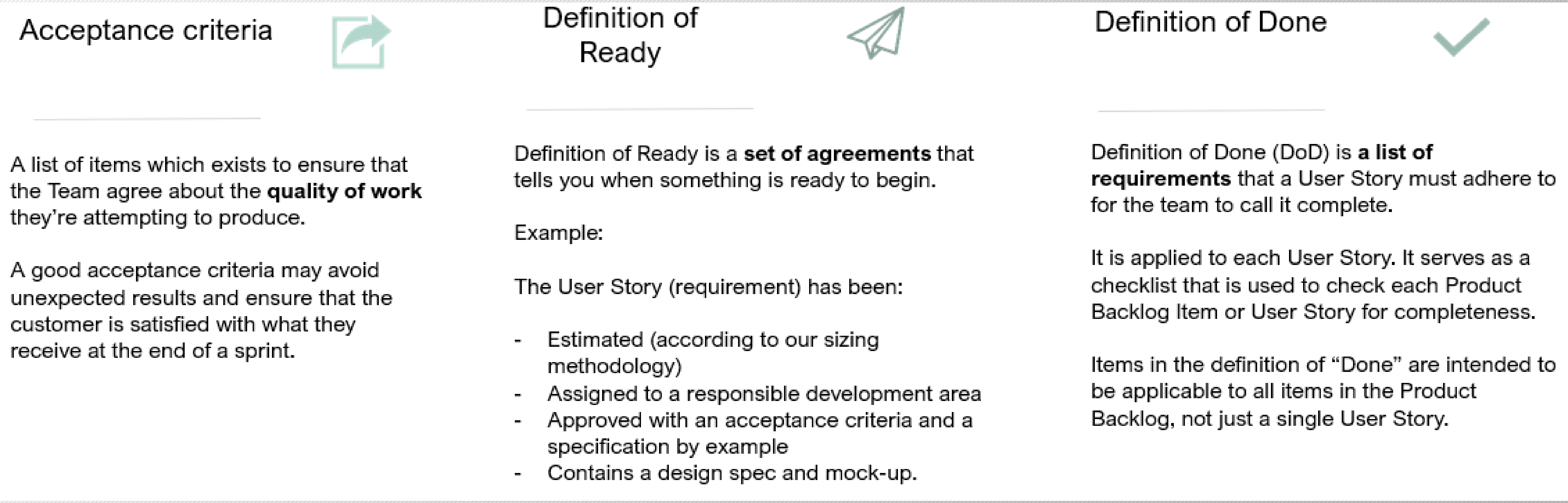
Product Increment

An Increment is the sum of all the Product Backlog items completed during a Sprint.

It adds up to the value of the increments of all previous Sprints.

Artifacts

Besides the artifacts, there are some elements to help increase the quality of the work and the collaboration within the team



Acceptance criteria

A list of items which exists to ensure that the Team agree about the **quality of work** they're attempting to produce.

A good acceptance criteria may avoid unexpected results and ensure that the customer is satisfied with what they receive at the end of a sprint.

Definition of Ready

Definition of Ready is a **set of agreements** that tells you when something is ready to begin.

Example:

The User Story (requirement) has been:

- Estimated (according to our sizing methodology)
- Assigned to a responsible development area
- Approved with an acceptance criteria and a specification by example
- Contains a design spec and mock-up.

Definition of Done

Definition of Done (DoD) is **a list of requirements** that a User Story must adhere to for the team to call it complete.

It is applied to each User Story. It serves as a checklist that is used to check each Product Backlog Item or User Story for completeness.

Items in the definition of "Done" are intended to be applicable to all items in the Product Backlog, not just a single User Story.

Agile Ceremonies

Sprint Planning

The product manager will prioritize the product backlog items the team will work on during that sprint

The product owner also discusses their initial plan for completing those product backlog items with the team

Daily Scrum

Daily Scrum is a short everyday meeting, ideally during start of the working day.

Each team member who works participates and updates on work completed the prior day, impediments and plan for the day.

Backlog refinement

Backlog refinement is when the product manager and some, or all, of the rest of the team review items on the backlog to ensure the backlog contains the appropriate items, prioritized by value.

The backlog refinement (sometimes called grooming) can happen any time in a sprint

Sprint Review & Demo

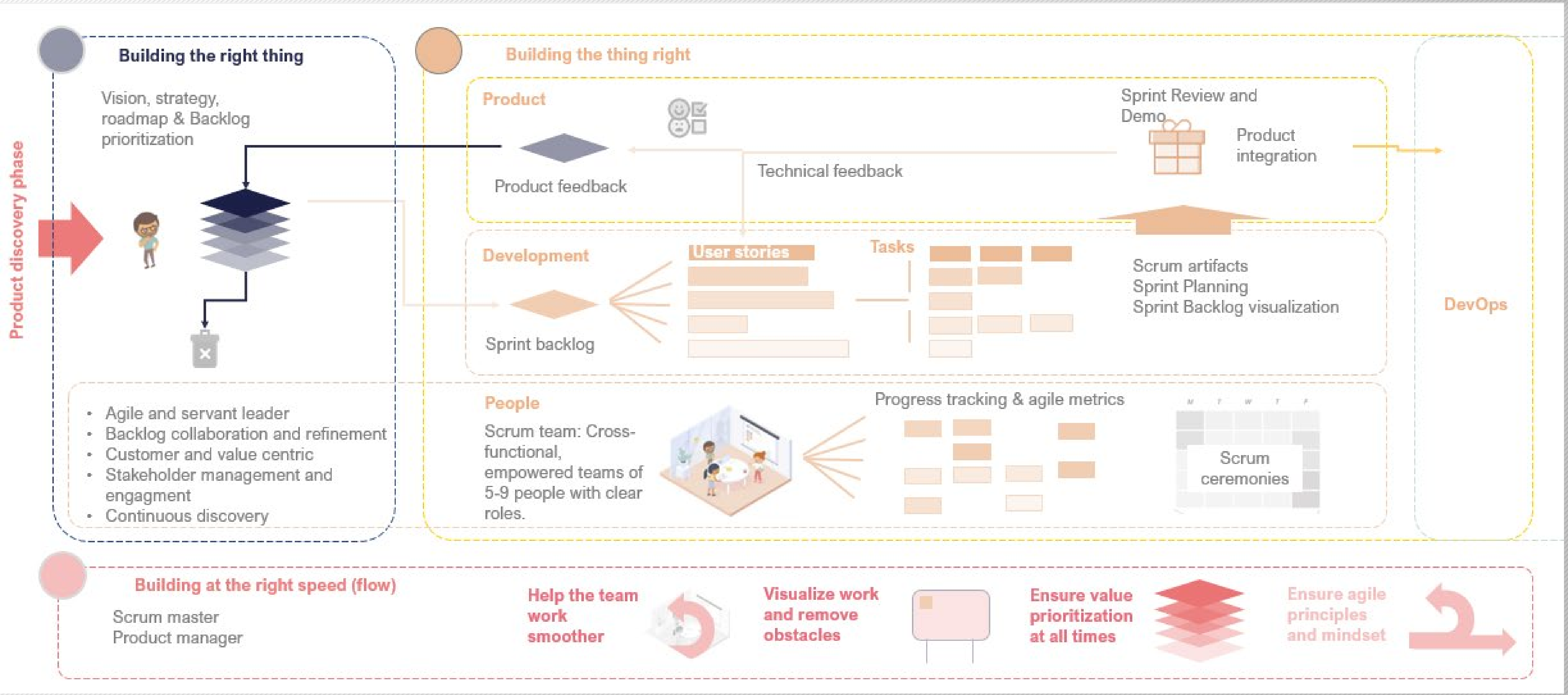
Sprint Review is a meeting which the team determine what are finished and what aren't & why.

The Demo is showcasing the value of work delivered to users and stakeholders.

Sprint Retrospective

Sprint Retrospective is a at which the team discusses the sprint and determines what could be change that might make the team continuously improve.

How Scrum Works: Overview



Product discovery phase

Building the right thing

Vision, strategy, roadmap & Backlog prioritization

- Agile and servant leader
- Backlog collaboration and refinement
- Customer and value centric
- Stakeholder management and engagment
- Continuous discovery

Building the thing right

Product

Product feedback

Technical feedback

Sprint Review and Demo

Product integration

Development

User stories

Tasks

Sprint backlog

Scrum artifacts
Sprint Planning
Sprint Backlog visualization

DevOps

People

Scrum team: Cross-functional, empowered teams of 5-9 people with clear roles.

Progress tracking & agile metrics

Scrum ceremonies

Building at the right speed (flow)

Scrum master
Product manager

Help the team work smoother

Visualize work and remove obstacles

Ensure value prioritization at all times

Ensure agile principles and mindset

What’s next...?