PMP AGILE IN A NUTSHELL

CONTEXT

WASHING
SORTING
CRUSHER
CONCENTRATE
JUICE
SHOP
OPEN

VUCA:

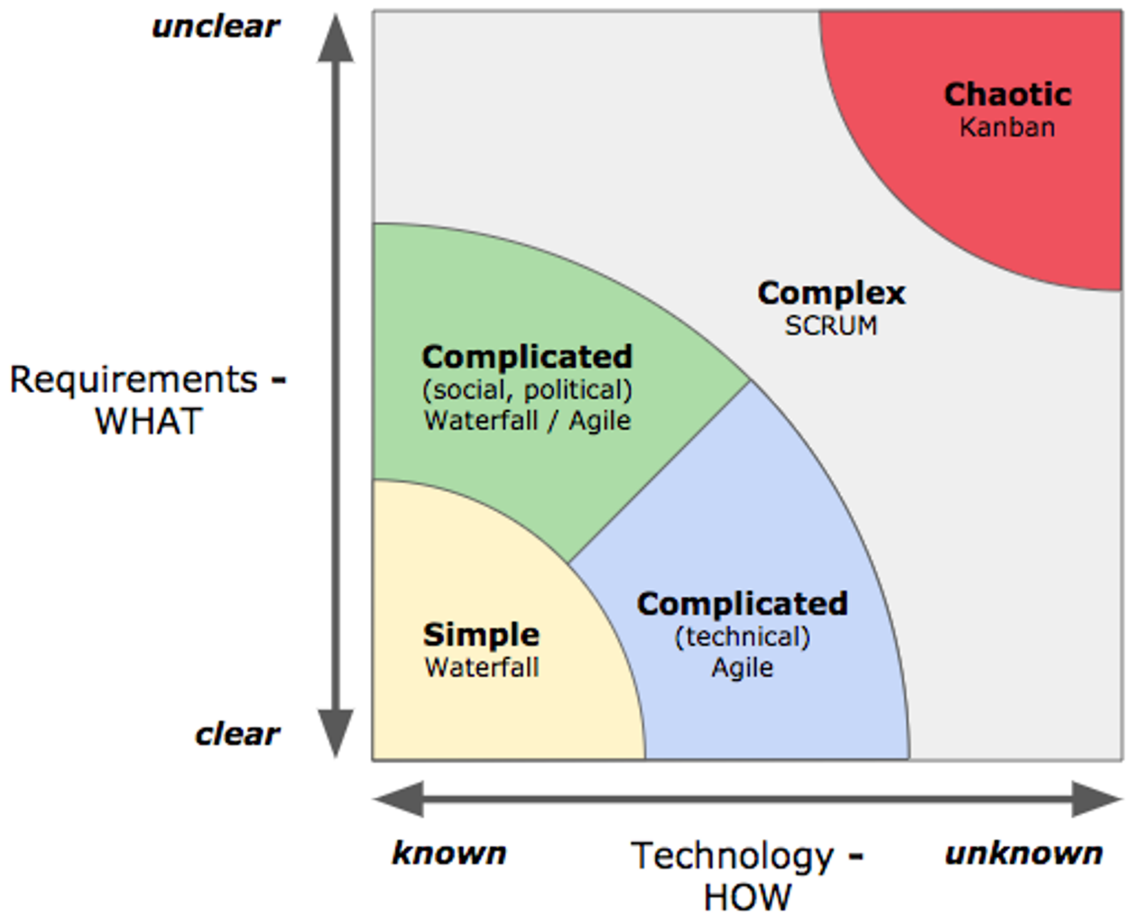
Volatility, Uncertainty, Complexity & Ambiguity

Definable work vs High – uncertainty work

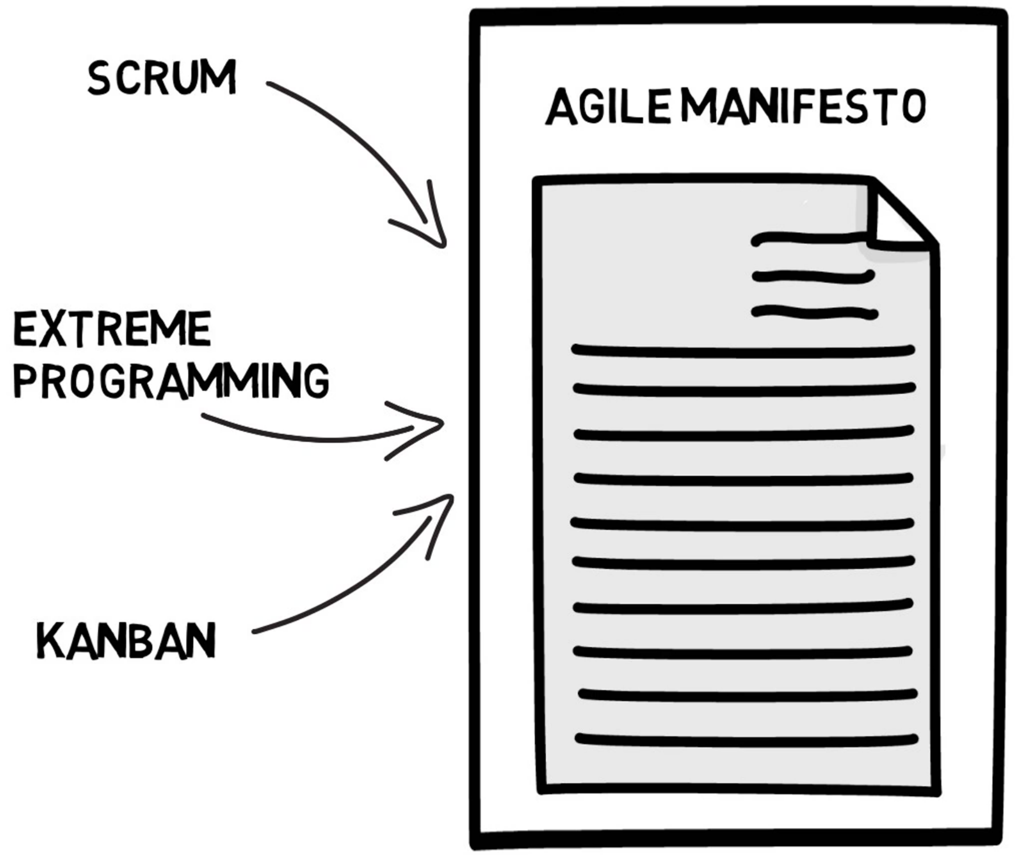
Definable work projects are characterized by clear procedures that have proved successful on similar projects in the past.

High-uncertainty projects have high rates of change, complexity, and risk.

Agile approaches were created to explore feasibility in short cycles and quickly adapt based on evaluation and feedback



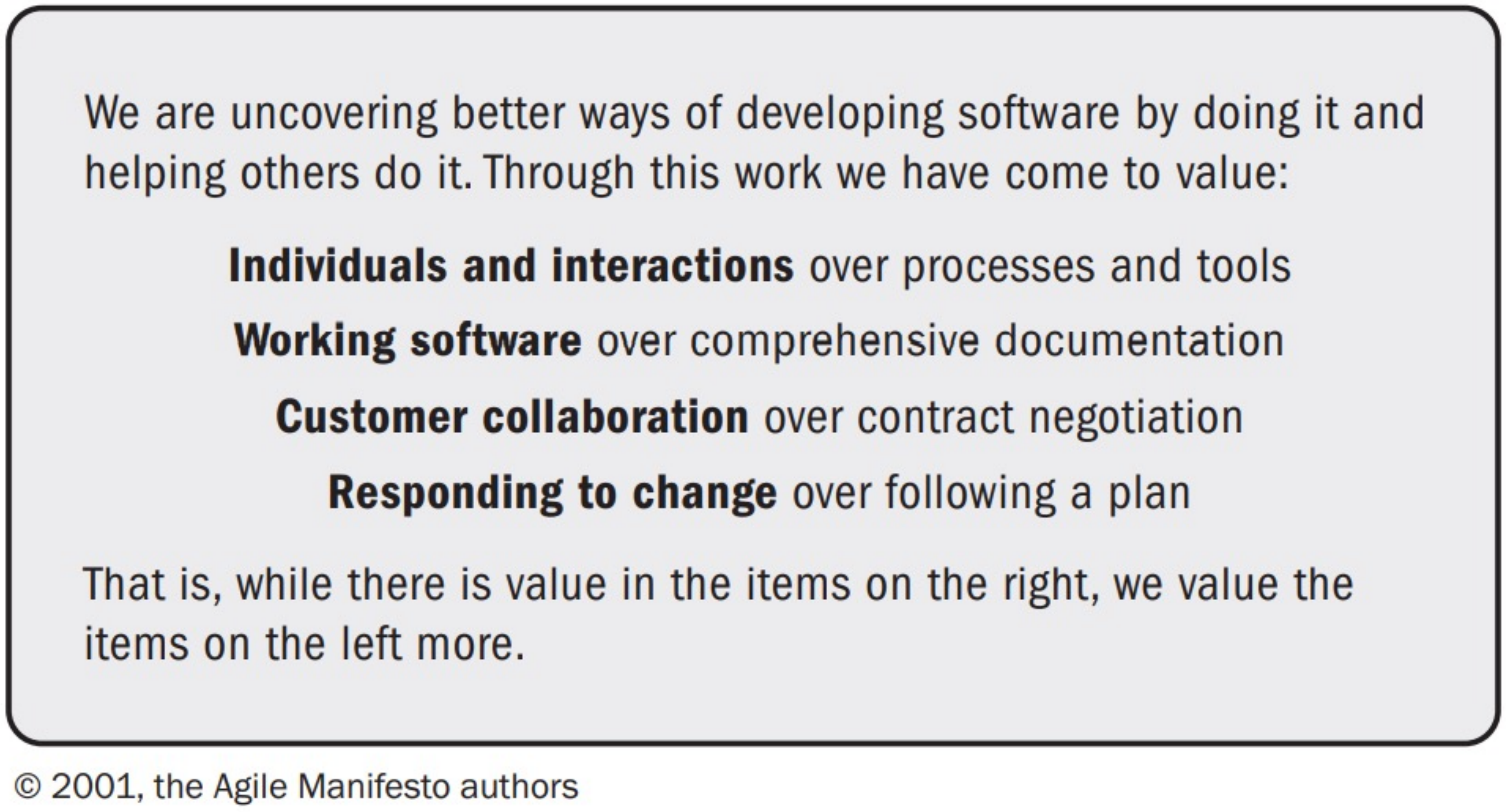
Agile Manifesto



Thought leaders in the software industry formalized the agile movement in 2001 with the publication of the Manifesto for Agile Software Development.

The values and ideas contained in this manifesto were derived from a larger range of software development frameworks including **Scrum, KanBan, Extreme Programming** and many others. So this is why Scrum is considered an agile framework and associated with the agile movement.

The Four Values



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 over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 2-1. The Four Values of the Agile Manifesto

The Four Values

Value 1: Individuals and Interactions Over Processes and Tools.

- We try to focus the team's attention on the individuals and interactions involved.
- Focusing early on developing the individuals involved in the project and emphasizing productive and effective interactions help setup a project success

Value 2: Working software Over Comprehensive Documentation

- We try to focus on the purpose or business value we're trying to deliver, rather than paperwork
- The agile approach to documentation
  - Just enough — only cover our needs; most of efforts focused on the emerging system
  - Just in time — don't have to spend extra time to keep it updated as reqs&designs change
  - Just in because — just produce it than to face the consequences of not doing

The Four Values

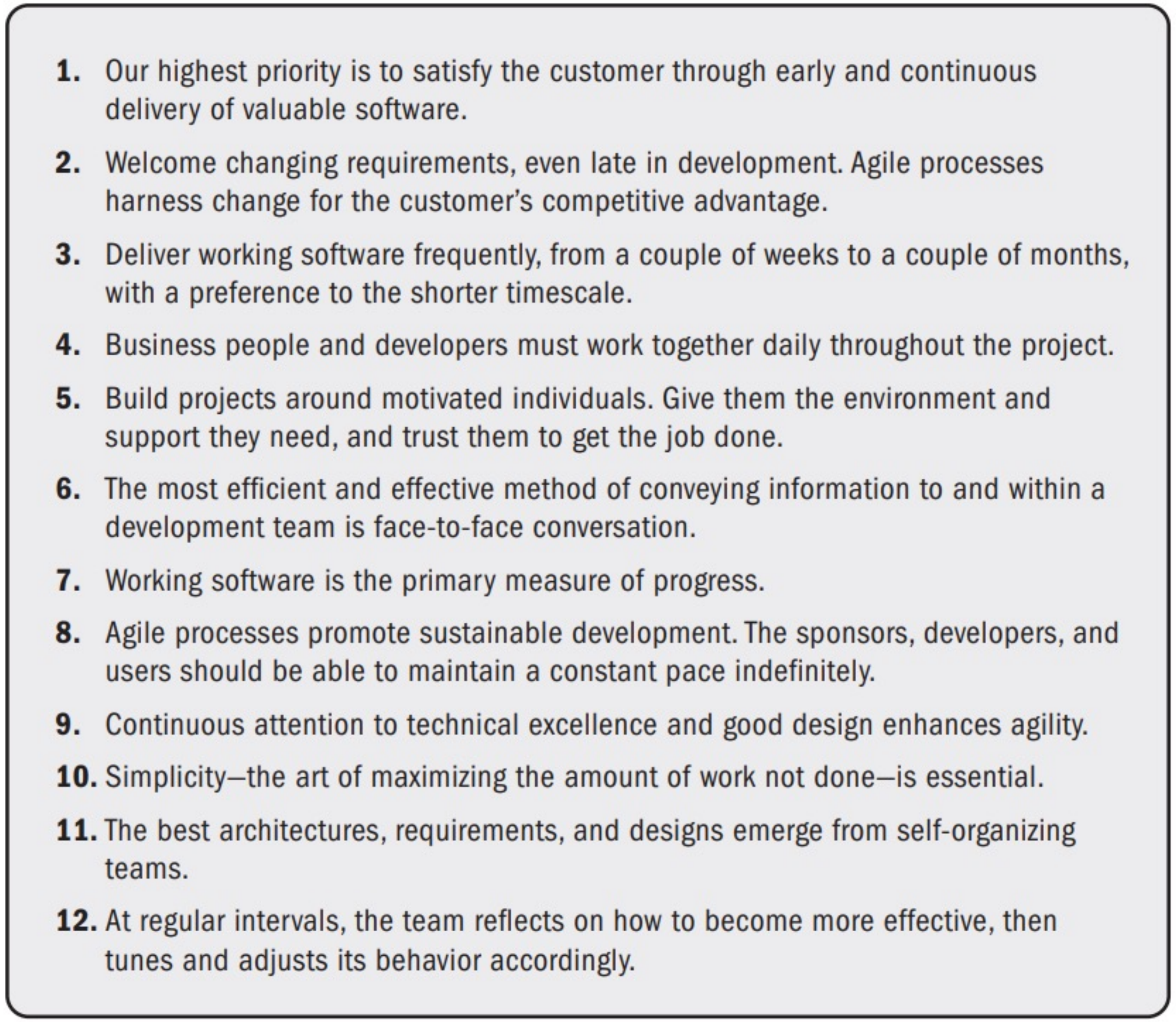
Value 3: Customer Collaboration Over Contract Negotiation.

- To be flexible and accommodating, rather than fixed and uncooperative.
- Requires a more trusting relationship and more flexible contract models; it moves the emphasis from non-value –adding activities to productive work.

Value 4: Responding to Change Over Following a Plan.

- Need to acknowledge initial plans were made when we knew least about the project and will need to be updated as the work progresses
- Agile projects have highly visible queues of work and plans in the form of backlogs and task boards. The intent of this value is to broaden the number of people who can readily engaged in the planning process by adjusting the plans and discussing the impact of changes

The twelve 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.
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 maximizing the amount of work not done—is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

The twelve Principles

Principle 1: Our highest priority is to satisfy the customer through early and continuous delivery of valuable software

- Our focus should be on the customer and make them satisfy by valuable software, not perfect plans and documentation
- It’s better to get something wrong up front and have time to correct it than to discover the issue much later when so much more has been built
- What we’re delivering is valuable software, not completed work products, WBS items, documentation, or plans

Principle 2: Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage

- Changes can be great for the project, for example, if they allow us to deliver a late-breaking, hight-priority feature
- Agile methods use a lightweight, high-visibility approach –for example, continuously updating and prioritizing changes into the backlog of work to be done . Agile’s well understood, high-visibility methods for handling changes keep the project adaptive and flexible.

The twelve Principles

Principle 3: Deliver working software frequently, from the couple of weeks to a couple of months, with a preference to the shorter timescale

- Principle emphasizes the importance of releasing work to a test environment and getting feedback
- Agile teams need feedback on what they have created thus far to see if they can proceed, or if a change of course is needed
- Delivering within a short timeframe also has the benefit of keeping the product owner engaged and keeping dialogue about the project going

Principle 4: Business people and developers must work together daily throughout the project

- Written documents, emails, and even telephone calls are less efficient ways of transferring information than face-to-face interactions
- Developers can learn about the business in a way that is far beyond what a collection of requirements-gathering meetings can ever achieve. As a result, development teams are able to suggest solutions and alternatives to business request. The business representatives also learn what types of solutions are expensive or slow to develop, and what features are cheap. They can begin to fine-tune their requests in response..

The twelve Principles

Principle 5: Build the projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

- Agile methods promote empowered teams. People work better when they are give the autonomy to organize and plan their our work.
- Agile methods advocate freeing the team from the micromanagement of completing tasks on a Gantt chart
- Knowledge work projects involve team members who have the unique areas of expertise. Such people do their best work when they are allowed to make many of day-to-day decisions and local planning for the project.

Principle 6: The most efficient and effective method of conveying information to and within a development team is face-to-face conversation

- Face-to-face conversation allow us to quickly transfer a lot of information in a richer way that includes emotions and body language
- In face-to-face conversation , questions can be immediately answered, instance of “parked” with the hope that there will be a follow-up explanation

The twelve Principles

Principle 7: Working software is the primary measure of progress

Shift focus to working results rather than documentation and design. In agile, we assess progress based on the emerging product or service we are creating.

The definition of progress as “working system” creates a results-oriented view of the project. Interim deliverables and partially completed work will get no external recognition. So we want to focus instead on the primary goal of the project - a product that delivers value to the business.

Principle 8: Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace of indefinitely

Agile methods strive to maximize value over the long term; Agile methods recognize the value of sustainable pace that allow team members to maintain a work-life balance. A sustainable pace is not only better for the team; it benefits the organization as well.

Working at a pace that can be maintained indefinitely leads to a happier and more productive team.

The twelve Principles

Principle 9: Continuous attention to technical excellence and good design enhance agility

We have to be mindful of keeping the design clean, efficient, and open to changes. Technical excellence and good design allow the development team to understand and update the design easily

Agile team needs to balance its efforts to deliver high-value features with continuous attention to the design of the solutions. This balance allows the product to deliver long-term value without becoming difficult to maintain, change, or extend.

Principle 10: Simplicity –the art of maximizing the amount of work not done –is essential

In software world, up to 60 percent of features that are built are used either infrequently or never. Because so many feature that are built are never actually used, and because complex system have an increased potential to be unreliable, agile methods focus on simplicity.

Agile methods seek the “simplest thing that could possible work” and recommend that this solution be built first . This approach not only mitigates risk but also helps boost sponsor confidence.

The twelve Principles

Principle 11: The best architectures, requirements, and designs emerge from self-organizing teams

- People like self-organizing; it allows them to find an approach that works best for their methods, their relationships, and their environment. They will thoroughly understand and support the approach, because they helped create it. As a result, they will produce better work.
- Self-organizing teams that have the autonomy to make local decisions have a higher level of ownership and pride in the architectures, requirements, and designs they create than in those that are forced on them or "suggested" by external sources
- The members of a self-organizing project team are closest to the technical details of the project. As a result, they are best able to spot implementation issues, along with opportunities for improvements

Principle 12: At regular intervals, the team reflects on how to become more effective, than tunes and adjusts its behavior accordingly

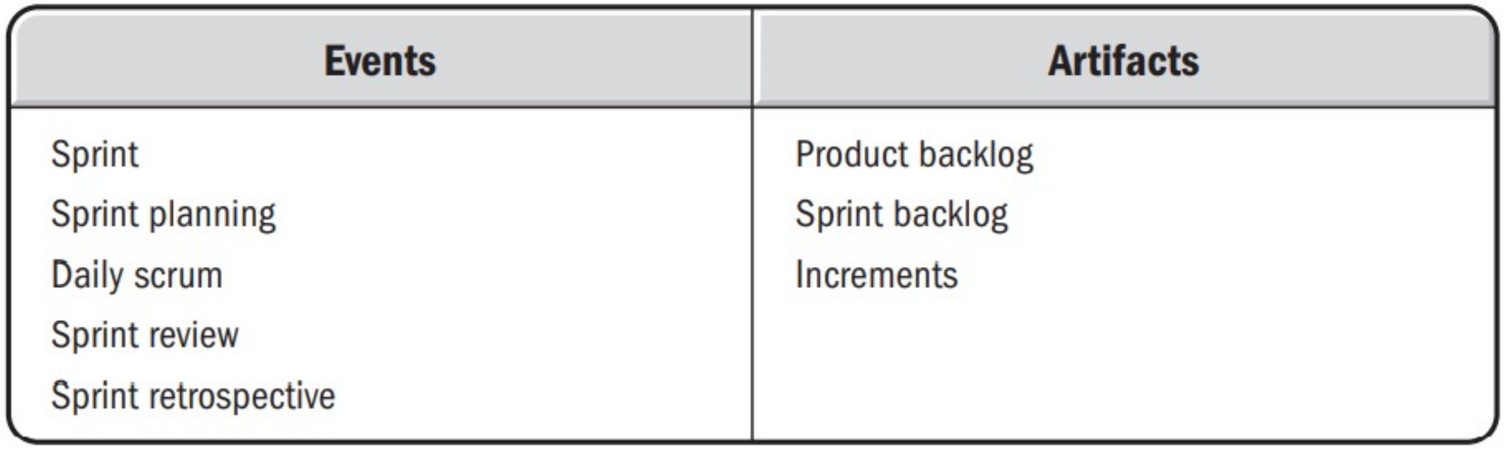
- Agile methods employ frequently lookbacks, call "restropectives", to reflect on how things are working on the project and identify opportunities for improvements.

Agile Methodologies

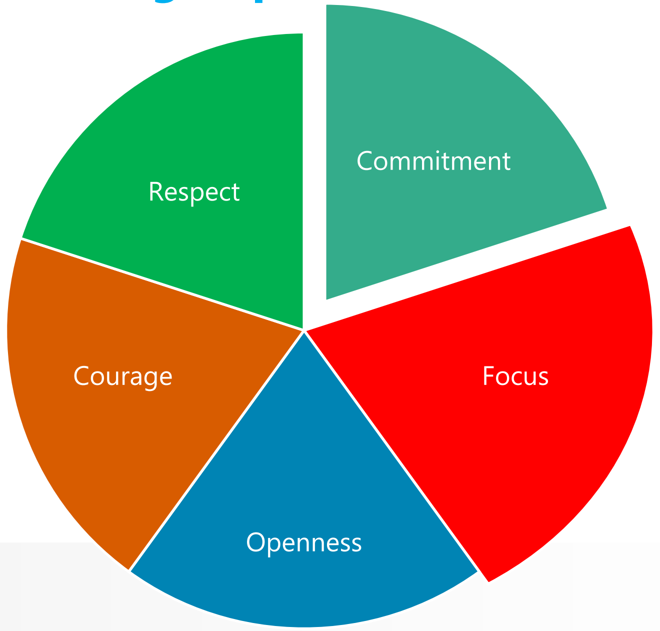
The most common approaches are Scrum, Extreme Programming (XP), Kanban, Feature-Driven Development (FDD), Dynamic Systems Development Method (DSDM), Scrumban, Agile Unified Process and the Crystal family of methods.

Scrum Framework

- Scrum is a framework which consist of roles, events, artifacts, and rules, and uses an iterative approach to deliver working product.
- Scrum is run on timeboxes of **1 month or less** with consistent durations called sprints where a potentially releasable increment of product is produced.
- The Scrum team consists of a product owner, development team, and scrum master.
- The **product owner** is responsible for maximizing the value of the product.
- The **development team** is a **cross-functional**, **self-organizing** team consisting of team members to deliver working product without depending on others outside of the team.
- The **scrum master** is responsible for ensuring the Scrum process is upheld and works to ensure the Scrum team **adheres to the practices and rules** as well as coaches the team on **removing impediments**.

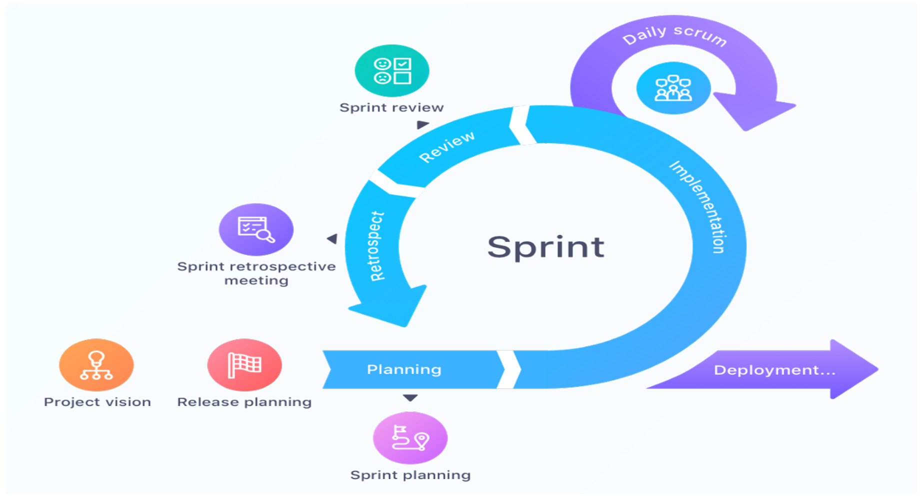


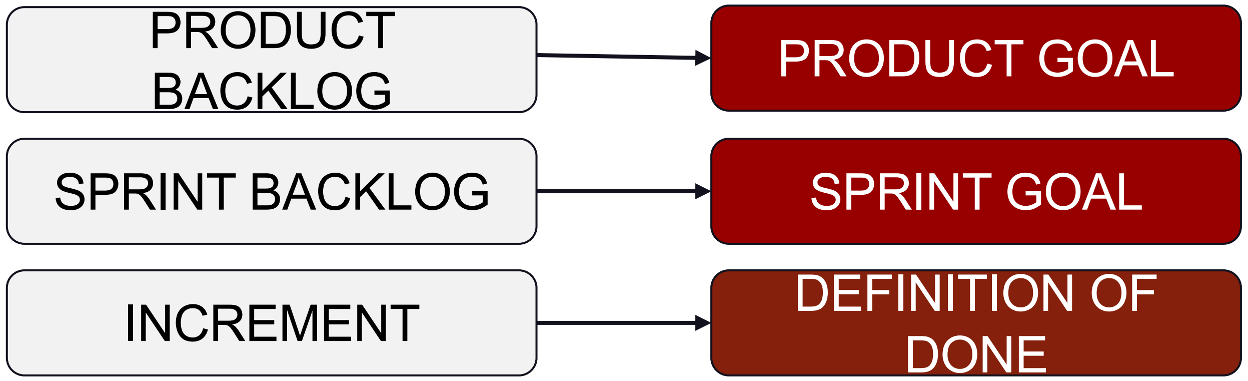
Events	Artifacts
Sprint	Product backlog
Sprint planning	Sprint backlog
Daily scrum	Increments
Sprint review	
Sprint retrospective	



Scrum Artifacts

- The Scrum guide defines three artifacts: **product backlog**, **sprint backlog**, **Increment.**
- Scrum artifacts represent **work or value**. They are designed to **maximize transparency** of key information. Thus, everyone inspecting them has the same basis for adaptation.
- Each of the Scrum artifacts contains a commitment to something
    - The **product backlog** exists to reach the long term objective - **product goal.**
    - The **sprint backlog** exists to reach the **sprint goal** that the Scrum team defines for every sprint. Each sprint goal is a step toward achieving the goal.
    - The **product increment** is committed to fulfilling the **definition of done**, which usually describes the desired quality that the product should have.





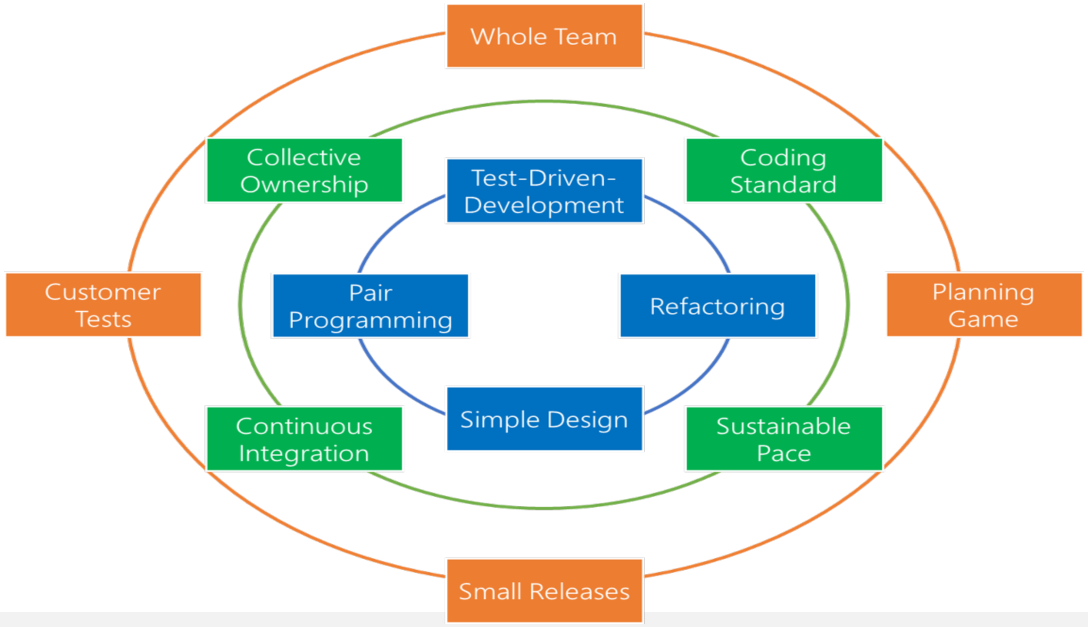
*A commitment means to be dedicated to achieving something specific. Having a commitment ensures that everyone knows why the work is important and what is the desired outcome.

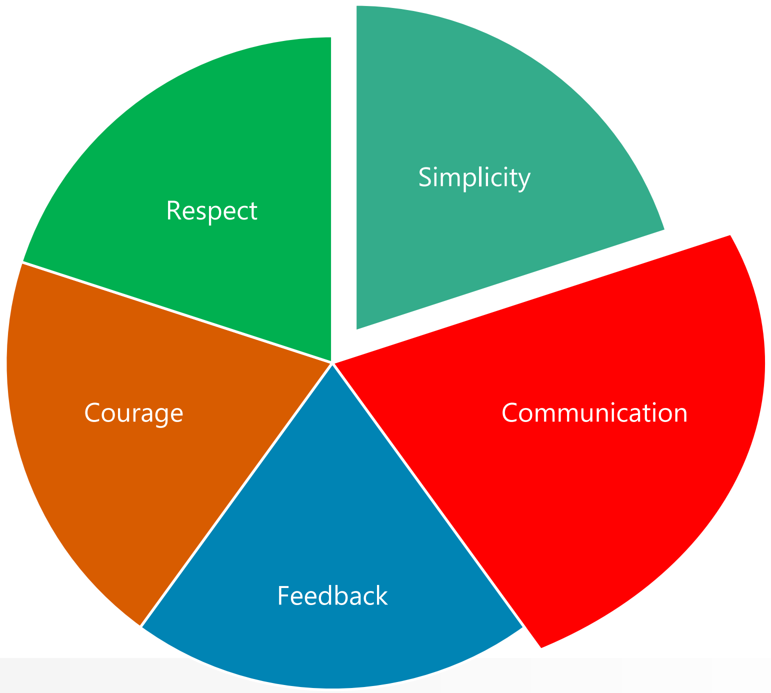
Scrum Events

- Scrum uses prescribed **events** or **meetings** or **ceremonies**, to reduce the need for other meetings that are not defined in Scrum.
- This does not mean that the Scrum team cannot have other meetings, but it is **mandatory** to have the Scrum events.
- At the heart of Scrum is the **sprint**, which acts as a container for all Scrum events. Remember that there are **no pauses or gaps between sprints** and everything happens within the sprint container.
- For this reason, the sprint is a special kind of an event. All events within Scrum have a maximum duration and are therefore - **Time-boxed**.
- Events are designed to **enable transparency, inspection and adaptation**. There are recommends that the events are held **at the same time and place** to create a routine and to **reduce complexity**.
  - **Sprint Planning**, where the work to be performed in the sprint is planned with no more than 8 hours time-boxed.
  - **The Daily Scrum**, which is held every day of the Sprint with no more than 15 minutes time-boxed.
  - **Sprint Review,** which is held at the end of the sprint to review the increment with no more than 4 hours time-boxed.
  - **Sprint Retrospective**, which is an opportunity to discuss ways to improve our all Scrum events, which are a formal opportunity to inspect and adapt with no more than 3 hours time-boxed.

Extreme Programming Framework

- **The coach** acts as a mentor to the team, guiding the process and helping the team members stay on track.
- On an XP team the "**customer**" is the business representative who provides the requirements, priorities, and business direction for the project.
- The **programmers** are the developers who build the product by writing and implementing the code for the requested user stories.
- The **testers** provide quality assurance and help the customer define and write acceptance tests for the user stories.



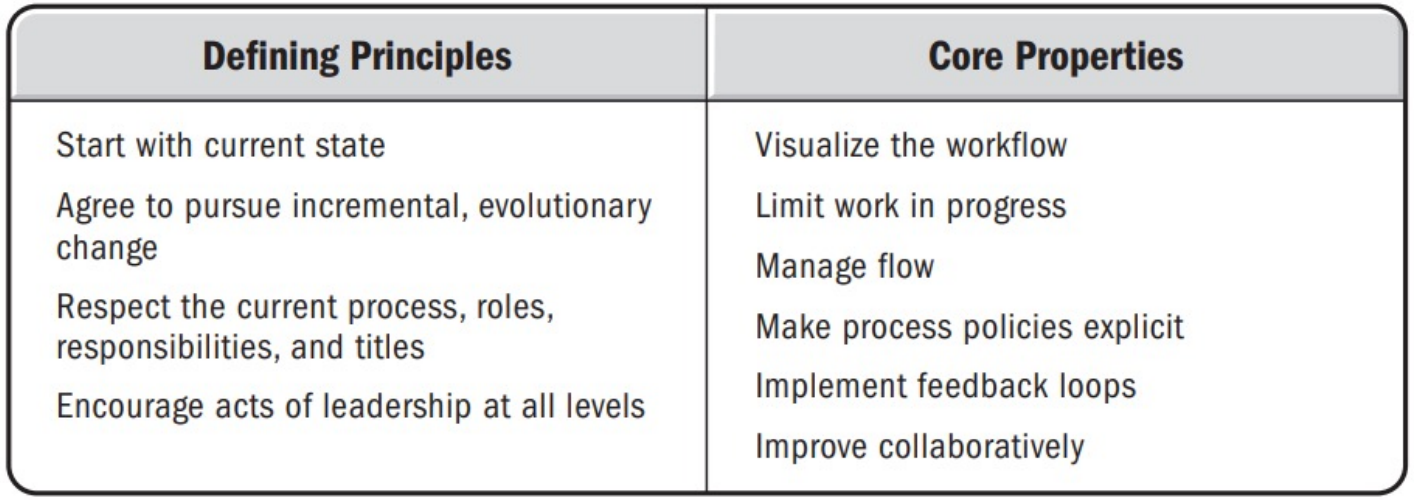


Extreme Programming Practices

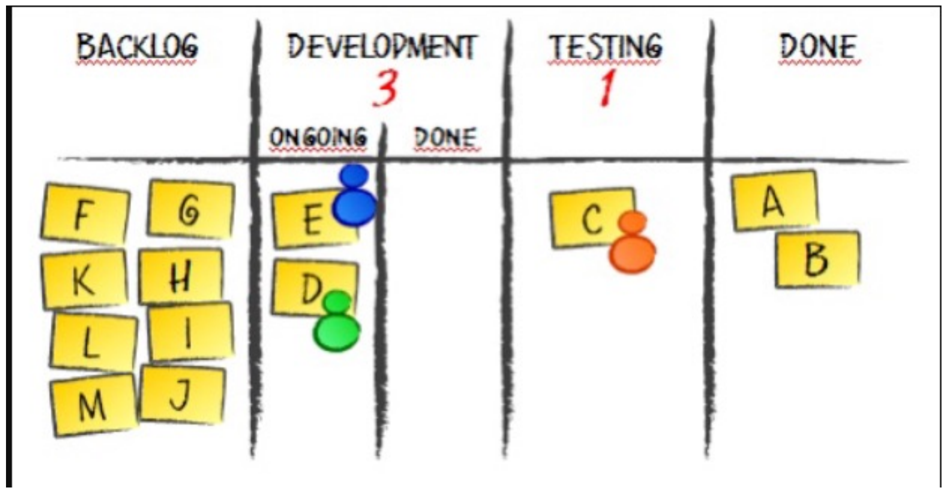
- **Whole team** sit together in the same location.
- Plan release and iteration use **planning games**.
- Frequent, **small releases** to a test environment are encouraged.
- **Customer describes** one or more customer test criteria that will indicate that the software working as intended.
- Multiple people work on the code to support **code collective ownership.**
- Follow a single **code standards** so that all the code look as if it has been written by a single knowledgeable programmer.
- Highest level of productivity is achieved by team operating at **sustainable pace**.
- **Metaphor** is used to explain design and creates shared technical vision.
- XP employs **continuous integration**, every time a programmer checks in code to the code repositor, integration tests are run automatically.
- Use **Test-Driven Development Approach** by writing the acceptance tests prior to developing the new code.
- Improving the design of existing code without altering its external behavior or adding new functionality through **refactoring**.
- By focusing on keeping the **design simple** but adequate ,XP teams can develop code quickly and adapt it as necessary.
- Production code is written by two developers working as a **pair**.

Kanban Method

- The Kanban Method is a holistic framework for **incremental, evolutionary process** and systems change for organizations.
- The method uses a **“pull system”** to move the work through the process.
- When the team completes an item, the team can pull an item into that step.
- Utilizing policies for entry and exit to columns, as well as constraints such as limiting **work in process**, Kanban boards provide clear insight to workflow, bottlenecks, blockers, and overall status.
- **Completing work** is more important than starting new work.

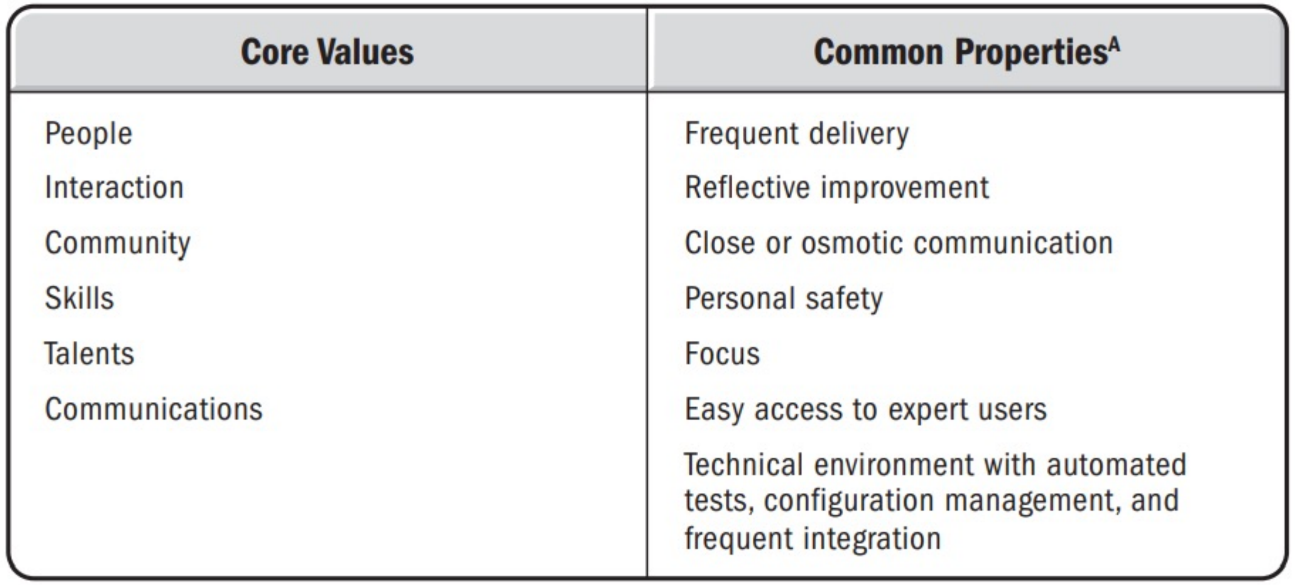


Defining Principles	Core Properties
Start with current state	Visualize the workflow
Agree to pursue incremental, evolutionary change	Limit work in progress
Respect the current process, roles, responsibilities, and titles	Manage flow
Encourage acts of leadership at all levels	Make process policies explicit
	Implement feedback loops
	Improve collaboratively

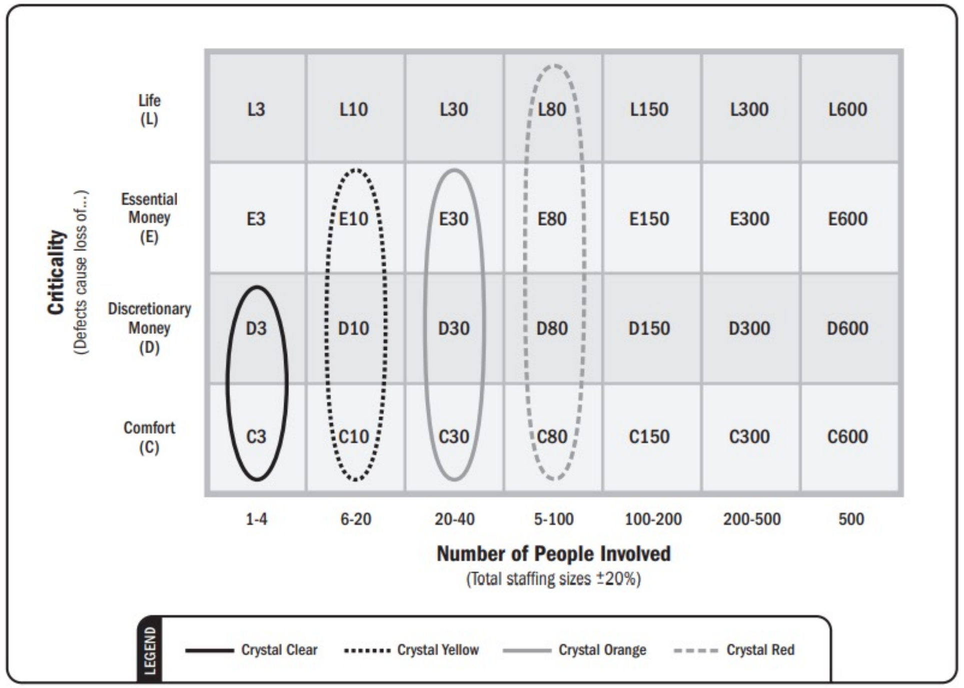


Crystal Method

- **A family** of situationally specific, customized methodologies that are coded by color names.
- Each methodology is customized by **criticality** and **team size**, which allows Crystal to cover a wide range of projects, from a small team building a **low-criticality system (Crystal Clear)** to a large team building a **high-criticality system (Crystal Red).**



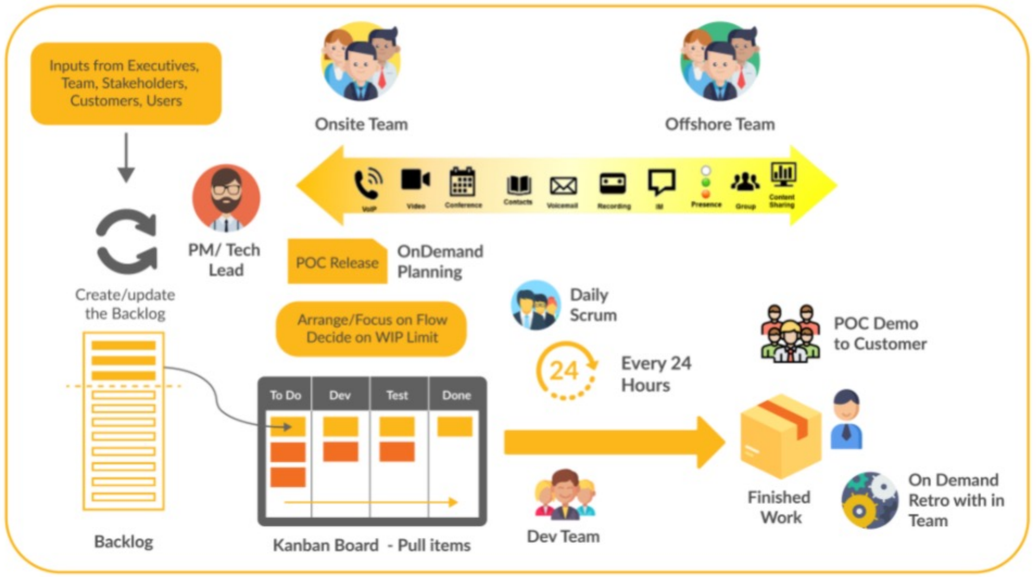
Core Values	Common Properties[A]
People	Frequent delivery
Interaction	Reflective improvement
Community	Close or osmotic communication
Skills	Personal safety
Talents	Focus
Communications	Easy access to expert users
	Technical environment with automated tests, configuration management, and frequent integration



Criticality (Defects cause loss of...)	1-4	6-20	20-40	5-100	100-200	200-500	500
Life (L)	L3	L10	L30	L80	L150	L300	L600
Essential Money (E)	E3	E10	E30	E80	E150	E300	E600
Discretionary Money (D)	D3	D10	D30	D80	D150	D300	D600
Comfort (C)	C3	C10	C30	C80	C150	C300	C600

Scrumban

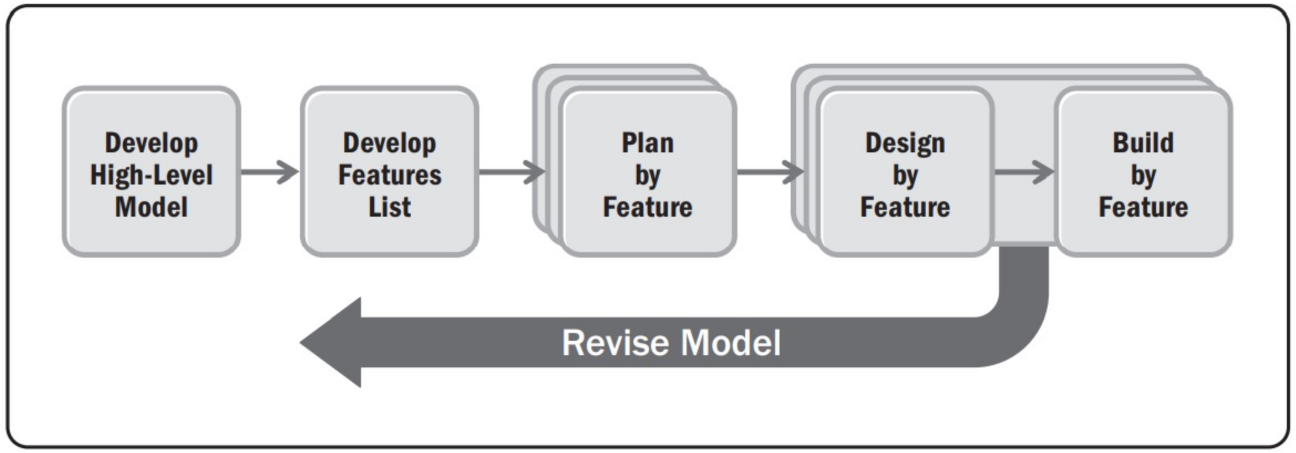
- **Scrumban** is an evolving hybrid framework in and of itself where teams use Scrum as a framework and Kanban for process improvement.
- Team organize the work into **small "sprints" and leverages the use of kanban boards** to visualize and monitor the work. The stories are placed on the kanban board and the team manages its work by using work-in-progress limits.
- **Daily meetings** are held to maintain the collaboration between the team and to remove impediments.
- There are **no predefined roles** in Scrumban—the team retains their current roles.



Feature Driven Development

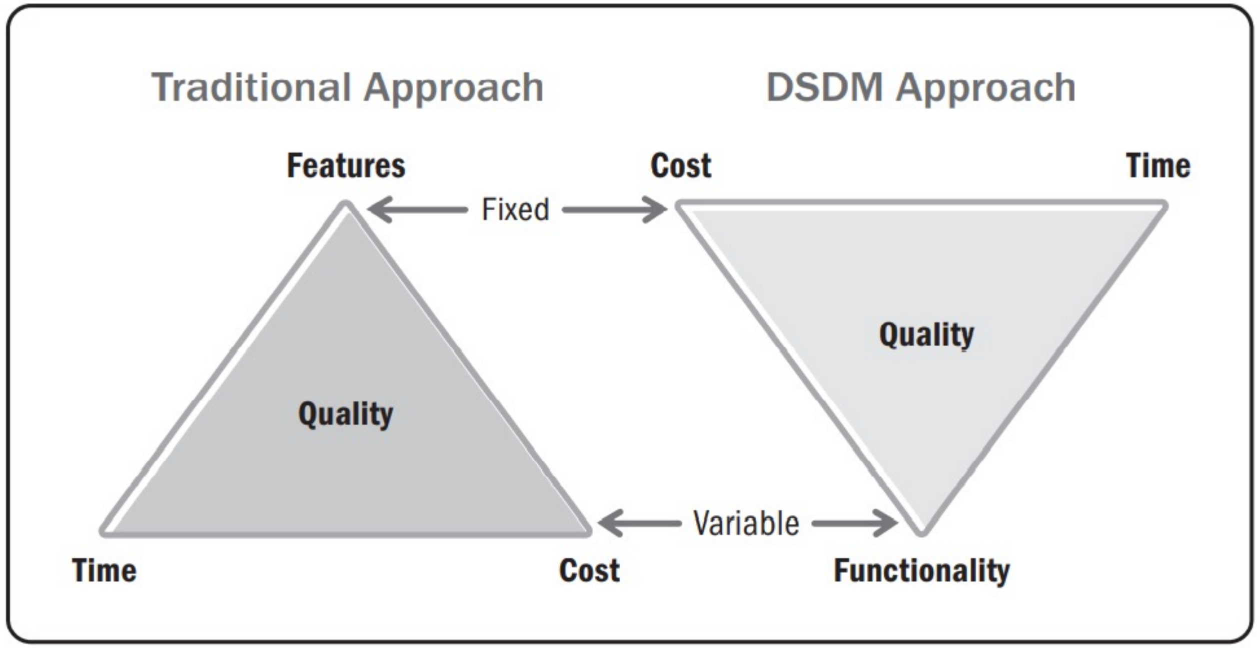
- **Feature-Driven Development (FDD)** was developed to meet the specific needs of a large software development project.
- There are six primary roles on a Feature-Driven Development project where individuals can take on one or more: **Project manager, Chief architect, Development manager, Chief programmer, Class owner, and/or Domain expert**.
- A Feature-Driven Development project is organized around five processes or activities: **develop an overall model, build a feature list, plan by feature, design by feature and build by feature**.
- Feature-Driven Development activities are supported by a core set of software engineering best practices:

- Domain object modeling
- Domain object modeling
- Individual class ownership
- Feature teams
- Inspections
- Configuration management,
- Regular builds
- Visibility of progress and results.



Dynamic Systems Development Method (DSDM)

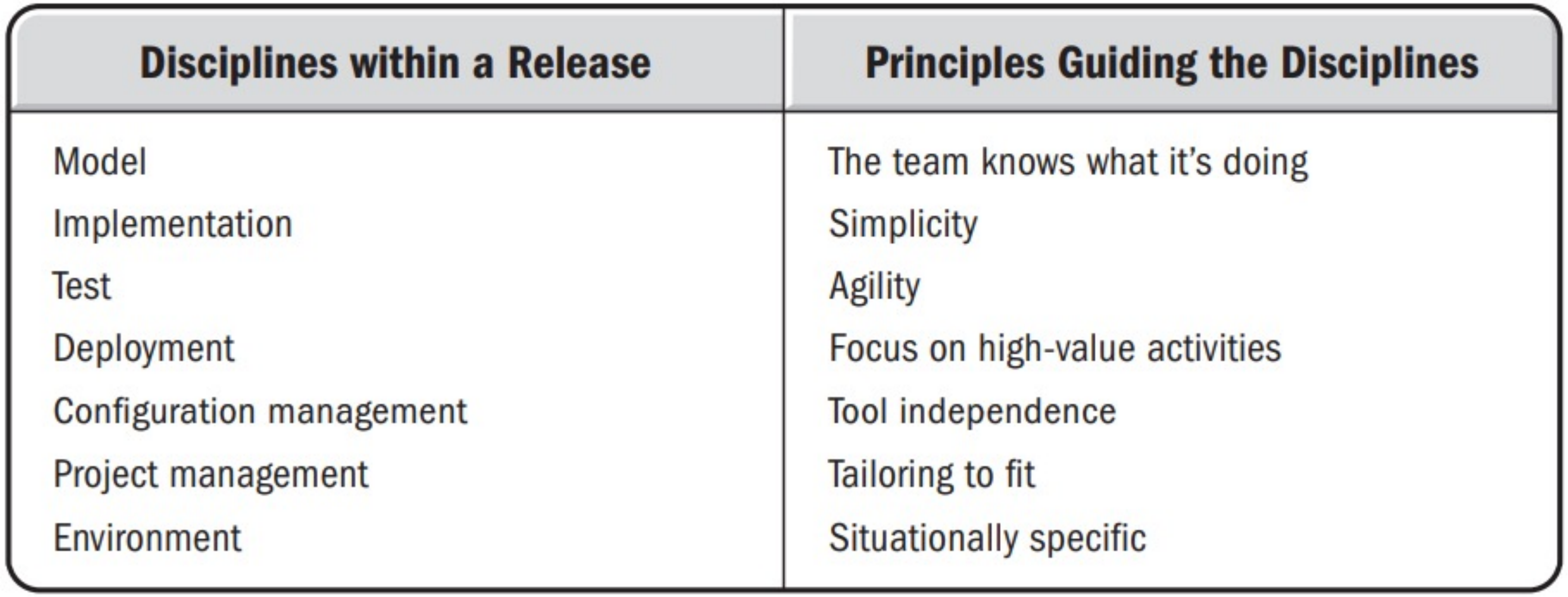
- **DSDM** is known best for its emphasis on **constraint-driven delivery**. The framework will set cost, quality, and time at the outset, and then use formalized prioritization of scope to meet those constraints.
- Eight principles guide the use of the DSDM framework:
  - Focus on the business need.
  - Deliver on time.
  - Collaborate.
  - Never compromise quality.
  - Build incrementally from firm foundations.
  - Develop iteratively.
  - Communicate continuously and clearly.
  - Demonstrate control



Agile Unified Process (AgileUP)

- AgileUP accelerated cycles and less heavyweight processes than its Unified Process predecessor.
- The intent is to perform more **iterative cycles** across seven key disciplines, and incorporate the associated feedback before formal delivery.

Table A3-5. The Key Elements of the Agile Unified Process



Disciplines within a Release	Principles Guiding the Disciplines
Model	The team knows what it's doing
Implementation	Simplicity
Test	Agility
Deployment	Focus on high-value activities
Configuration management	Tool independence
Project management	Tailoring to fit
Environment	Situationally specific

Questions

Which of the following is an Agile Manifesto value?.

a. Individuals and interactions over following a plan.

b. Working software over processes and tools

c. Responding to change over comprehensive documentation.

d. Customer collaboration over contract negotiation.

D. The third value of the Agile Manifesto is "Customer collaboration over contract negotiation:' While the other choices use terms from the other three agile values, they aren't combined correctly.."

Questions

The agile triangle of constraints is said to be inverted from the traditional triangle because it allows:.

a. Scope and time to vary instead of cost.

b. Cost and time to vary instead of scope.

c. Scope and cost to be fixed instead of time.

d. Scope to vary while time and cost are fixed.

D. Unlike the traditional constraint triangle, in which scope is fixed and time and cost may need to bend to achieve that planned scope, agile teams typically allow scope to vary within fixed parameters of cost and time. In other words, they aim to deliver the most value they can by X date within X budget.

Questions

Which of the following Agile Manifesto principles reflects the agile focus on team empowerment?.

a. Working software is the primary measure of progress.

b. Welcome changing requirements, even late in development.

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

d. Build projects around motivated individuals.

D. Agile Manifesto principle five, "Build projects around motivated individuals" addresses the importance of giving teams the environment and support they need, and trusting them to get the job done. Supporting and trusting the team members means recognizing that they are experts at what they do, and that they can work most effectively if they are empowered to plan and organize their own work.

Questions

Which of the following isn't a core aspect of the agile mindset?.

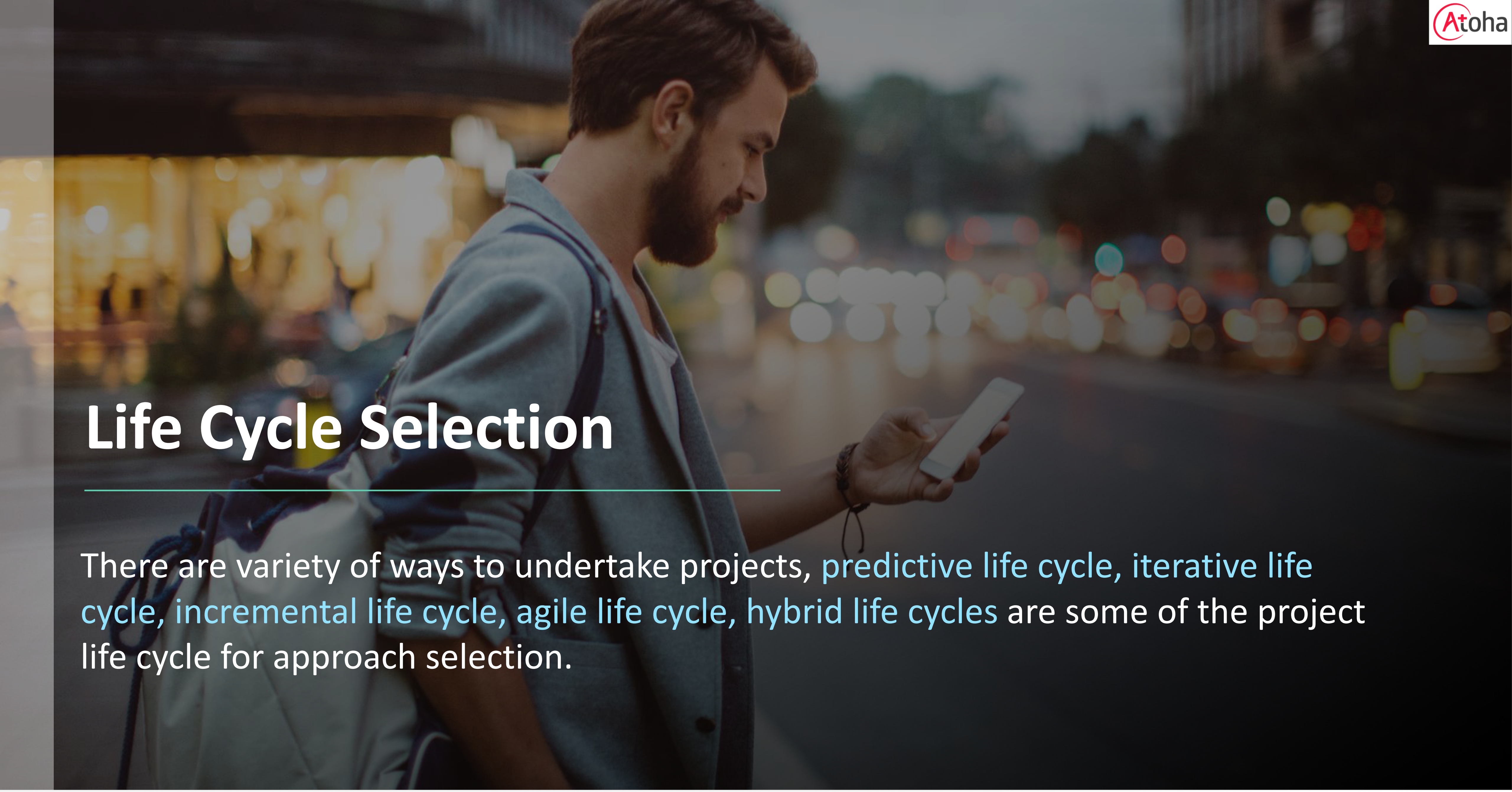
a. Welcome change.

b. Learn through discovery.

c. Respect the process.

d. Deliver value continuously.

C. The first value of the Agile Manifesto-"individuals and interactions over processes and tools"-is another indication that "respect the process" isn't part of the agile mindset.

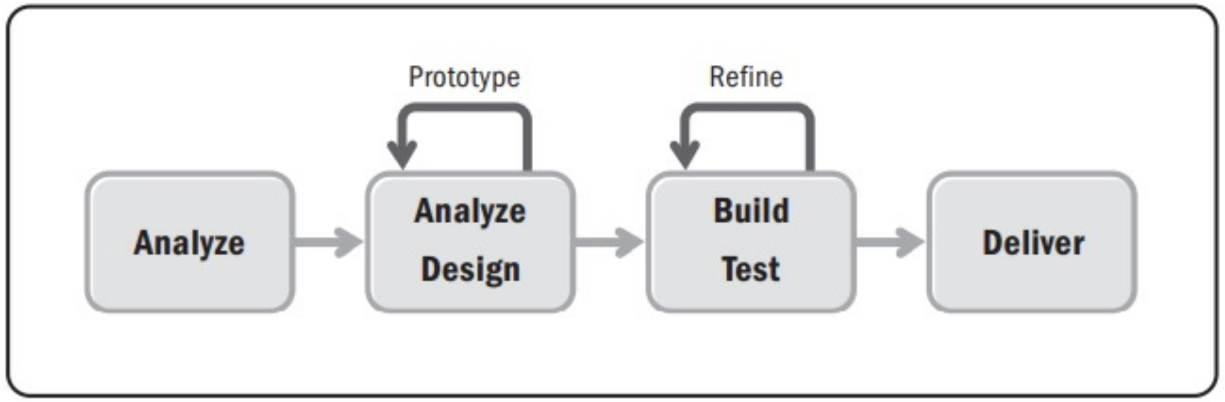


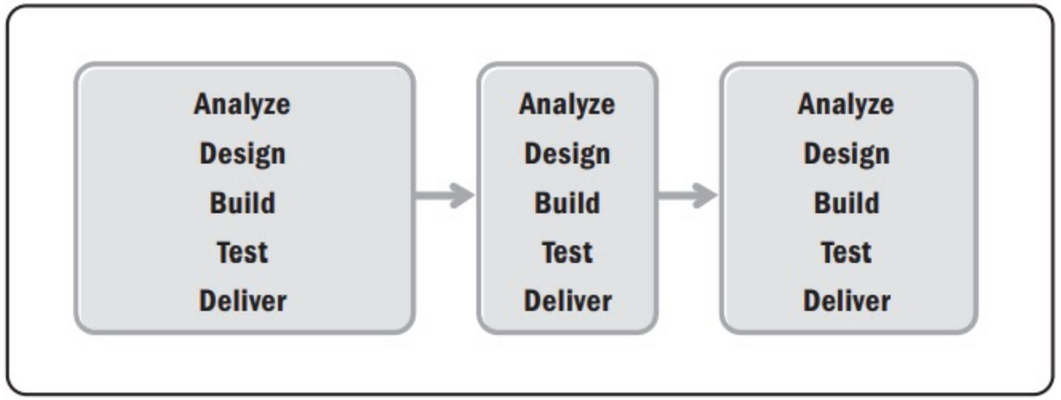
Life Cycle Selection

There are variety of ways to undertake projects, predictive life cycle, iterative life cycle, incremental life cycle, agile life cycle, hybrid life cycles are some of the project life cycle for approach selection.

Incremental and Iterative

Iterative Life cycles	Incremental life cycle
• Successive prototypes or proofs of concept to improve product results. • Iterations help identify and reduce uncertainty in the project. • Iterative life cycles may take longer because they are optimized for learning rather than speed of delivery	• Incremental life cycle deliver a subset of the overall solution to optimize for speed of delivery.

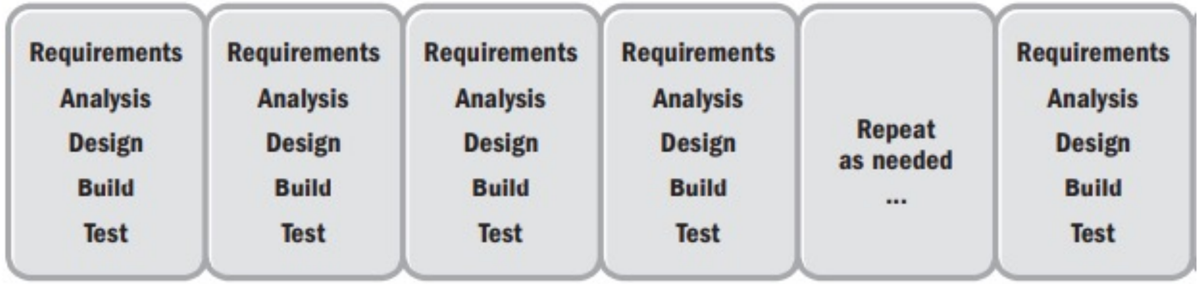




Agile Life Cycle

Iteration-based Agile	Flow-based Agile
• The team works in iterations (timeboxes of equal duration) to deliver completed features. • The team works on the most important feature, collaborating as a team to finish it. Then the team works on the next most important feature and finishes it.	• The team pulls features from the backlog based on its capacity to start work. • The team defines its workflow with columns on a task board and manages the work in progress • Each feature may take a different amount of time to finish. • Teams keep work-in-progress sizes small to better identify issues early and reduce rework. • The team and business stakeholders determine the most appropriate schedule for planning, product reviews, and retrospectives.

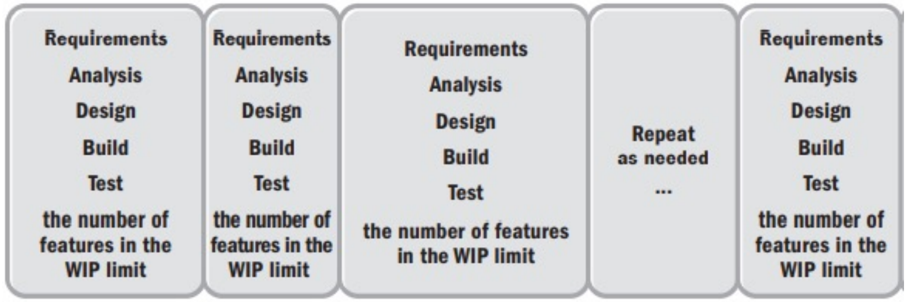
Iteration-Based Agile



Requirements Analysis Design Build Test
Requirements Analysis Design Build Test
Requirements Analysis Design Build Test
Requirements Analysis Design Build Test
Repeat as needed ...
Requirements Analysis Design Build Test

NOTE: Each timebox is the same size. Each timebox results in working tested features.

Flow-Based Agile



Requirements Analysis Design Build Test the number of features in the WIP limit
Requirements Analysis Design Build Test the number of features in the WIP limit
Requirements Analysis Design Build Test the number of features in the WIP limit
Repeat as needed ...
Requirements Analysis Design Build Test the number of features in the WIP limit

NOTE: In flow, the time it takes to complete a feature is not the same for each feature.

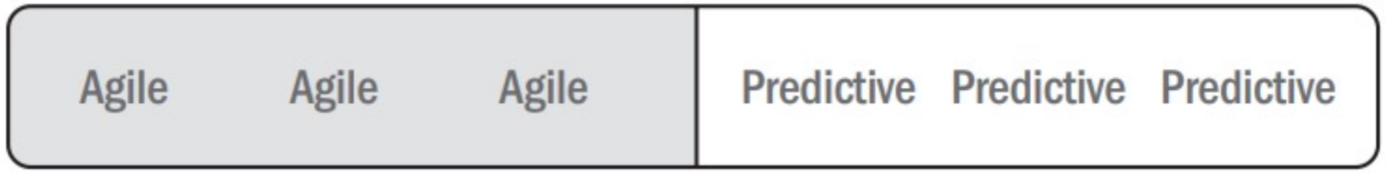
Atoha

Hybrid Life Cycles

A **combination of predictive, iterative, incremental**, and/or **agile approaches** is a **hybrid approach**.

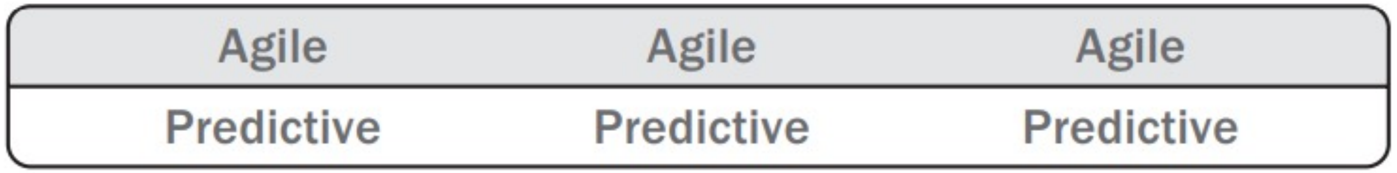
Approach 1. Early processes utilize an agile development life cycle, which is then followed by a predictive rollout phase.

Case. The development portion of the project might suitable for agile approach while the repeatable rollout phase that is appropriate to undertake in a predictive way



Approach 2. Combination of agile and predictive approaches throughout the life cycle.

Case. The team is incrementally transitioning to agile and using some agile practices but other aspects of the project such as upfront estimation, work assignment, and progress tracking are still following predictive approaches

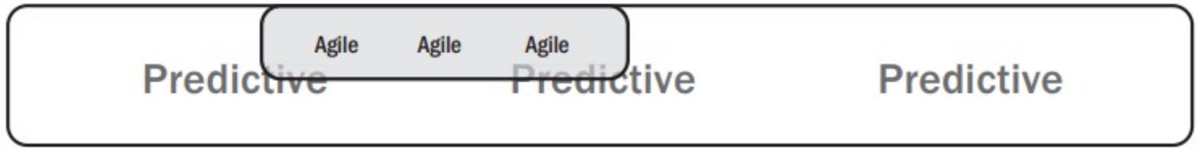


Hybrid Life Cycles

A **combination of predictive, iterative, incremental**, and/or **agile approaches** is a **hybrid approach**.

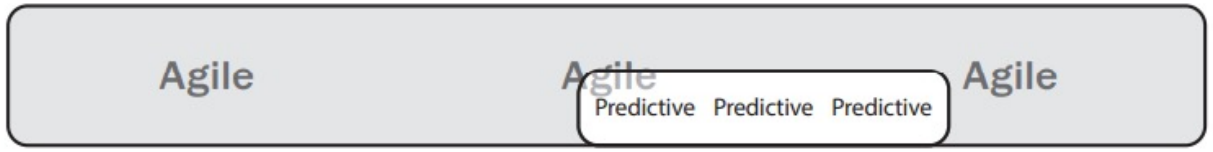
Approach 3. Largely Predictive Approach with Agile Components

Case. A portion of the project with uncertainty, complexity, or opportunity for scope creep is being tackled in an agile way, but the remainder of the project is being managed using predictive approaches.

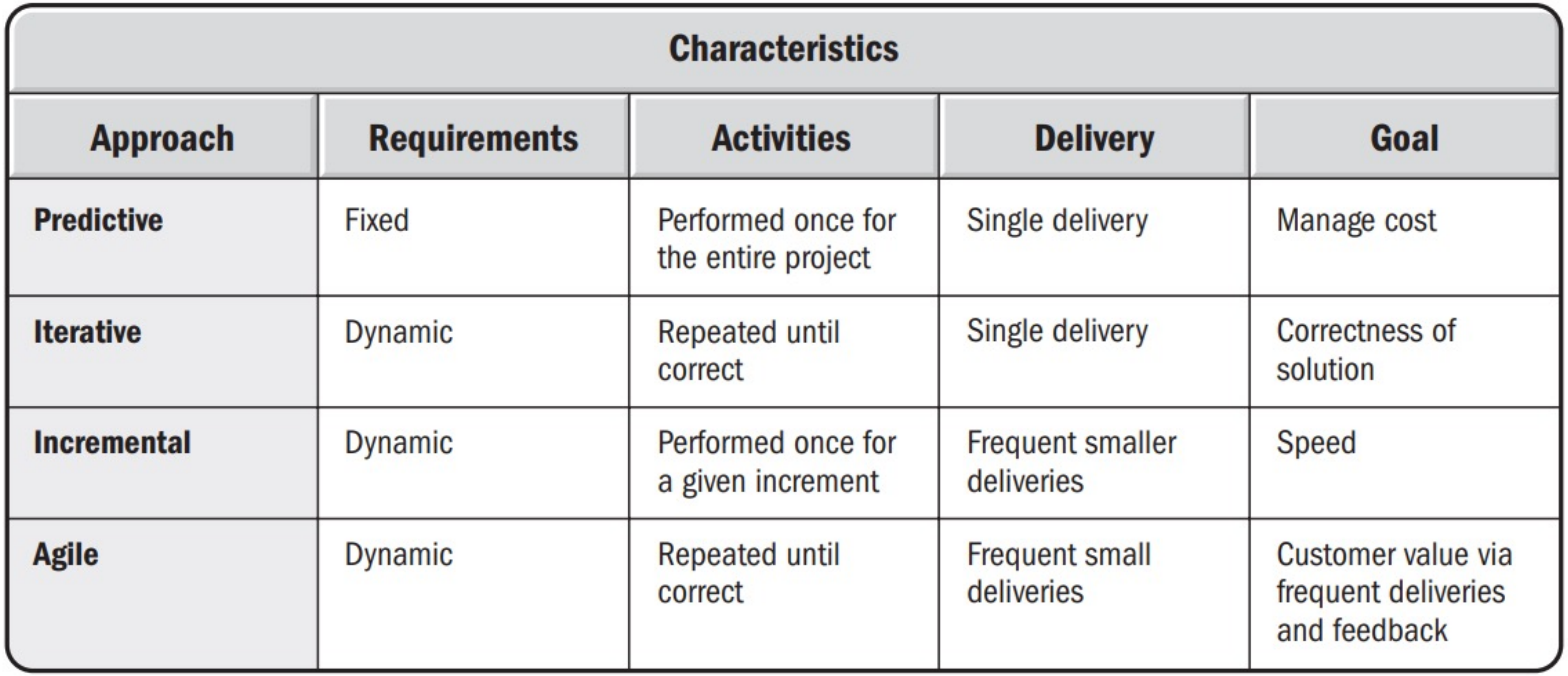


Approach 4. A Largely Agile Approach with a Predictive Component

Case. A particular element is non-negotiable or not executable using an agile approach.



Characteristic of Life Cycles



Characteristics				
Approach	**Requirements**	**Activities**	**Delivery**	**Goal**
Predictive	Fixed	Performed once for the entire project	Single delivery	Manage cost
Iterative	Dynamic	Repeated until correct	Single delivery	Correctness of solution
Incremental	Dynamic	Performed once for a given increment	Frequent smaller deliveries	Speed
Agile	Dynamic	Repeated until correct	Frequent small deliveries	Customer value via frequent deliveries and feedback

Questions

You are analyzing a process and are on the lookout for the anti-patterns of poor methodologies. What characteristics should you be looking for?

Select one:

a. One-of-a-kind, disciplined, heavy, embellished

b. One-size-fits-all, disciplined, heavy, embellished

c. One-size-fits-all, intolerant, heavy, embellished

d. One-of-a-kind, intolerant, embellished.

C. One-size-fits-all, intolerant, heavy, embellished. If a methodology is one-of-a-kind, this might also be a warning sign, since it means the methodology has not been repeated, but it is not as concerning as a claim that the methodology is a one-size-fits-all approach. Such a claim demonstrates a lack of situational awareness. The characteristic of being disciplined is not something to be wary of, since agile methods are very disciplined; we should not mistake being disciplined for being process-heavy

Questions

Your sponsor is asking about tailoring the company's newly adopted agile methodology. Your advice should be:

Select one:

a. Tailoring it will be a good way to learn more about the methodology.

b. Tailoring it will be a good way to ease into the initial adoption process.

c. We should tailor it first, then consider adopting it.

d. We should try it first, then consider tailoring it.

D. Agile methods should be tried as-is first before considering modifications for process tailoring. We needs to first understand how the practices work before we attempt to change them. If we change the method first and then encounter problems, how will we know if the problems are genuine project issues or the result of the changes we made?

Agile Team

Describes all the roles form an agile team and their responsibilities as well as servant leadership's duties.

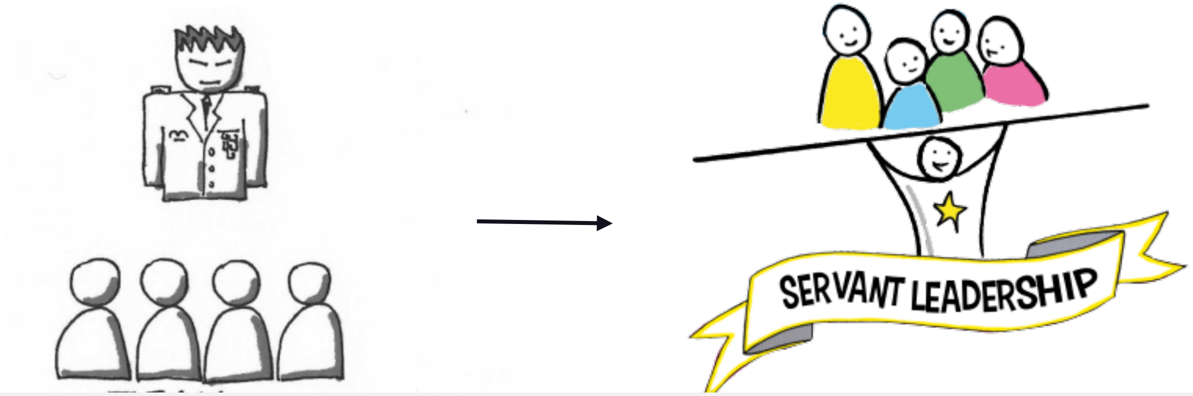
Servant Leadership

Servant leadership is the practice of leading through service to the team, by focusing on understanding and addressing the needs and development of team members in order to enable the highest possible team performance. Servant Leadership work with the team to define according to the following order:

Purpose. Work with the team to define the “why” or purpose so they can engage and coalesce around the goal for the project.

People. Once the purpose is established, encourage the team to create an environment where everyone can succeed.

Process. Do not plan on following the “perfect” agile process, but instead look for the results. When a cross functional team delivers finished value often and reflects on the product and process, the teams are agile.



Four Duties of a Servant Leadership

① Shield the Team From Interruptions

- Isolate and protect the team.
- Keep business partners from interrupting the team design.
- Ensuring people communicate to the designated channels.
- Protect the team from the diversions.

② Remove Impediments To Progress

- Clear obstacles for the team.
- Remove things that may cause delays or waste.
- Consider documentation compliance activities or anything that does not directly add value.
- Daily scrum give insights for the servant leader.

③ Communicate The Project Vision

- Ensures that stakeholders have a clear image of what the team is creating.
- Ensures that all stakeholders have a common vision of what done means.
- Communicate and recommunicate the project Vision to reinforce the vision.

④ Carry Food and Water

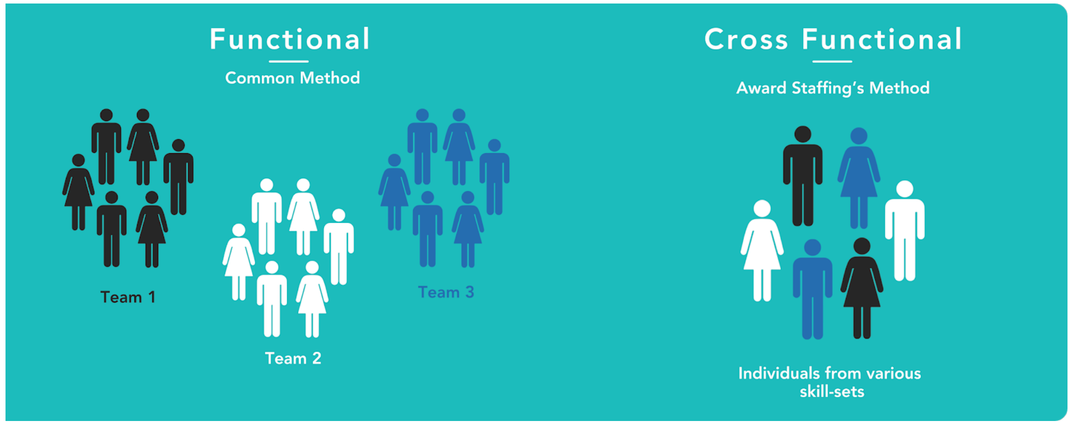
- Ensuring that the team has the resources they need to be productive
- This includes items like proper tools compensation encouragement and other resources
- Training and professional development may be included as well

Agile Team

- The most effective agile teams tend to range in size from **three** to **nine members**.
- Agile teams are **co-located** in a team space or has the ability to manage any location challenges.
- Team members are 100% **dedicated** to the teams to increase focus and productivity.
- Agile encourages **self-managing** teams, where team members decide who will perform the work within the next period's defined scope.
- Agile teams thrive with **servant leadership**.
- **Cross-functional** agile teams produce functional product increments frequently. Teams are mixed of generalist and specialist.
- Agile team maintains a **stable work environment** to reach the highest productivity.
- There are three common roles in agile: **cross-functional team members**, **product owner** and **team facilitators**.

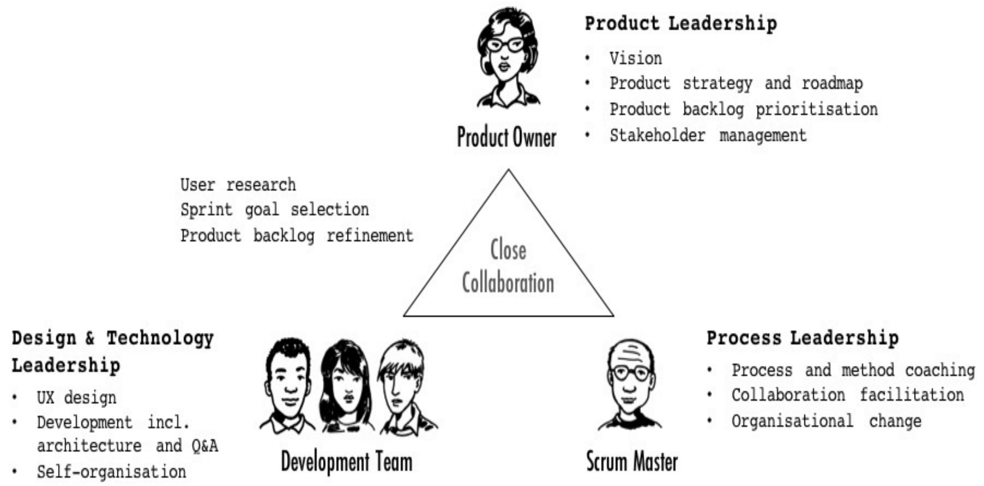
Cross-functional team members

- **Cross-functional teams** consist of team members with all the skills necessary to produce a working product.
- For examples, they can be designers, developers, testers, and any other required roles.
- The cross-functional team members deliver **potentially releasable product** on a regular cadence.
- Cross-functional teams are critical because they can deliver finished work in the shortest possible time, with higher quality, without external dependencies.
- In Scrum framework, they can be called as **developers**, while XP describes these roles as **programmers**.



Product Owner

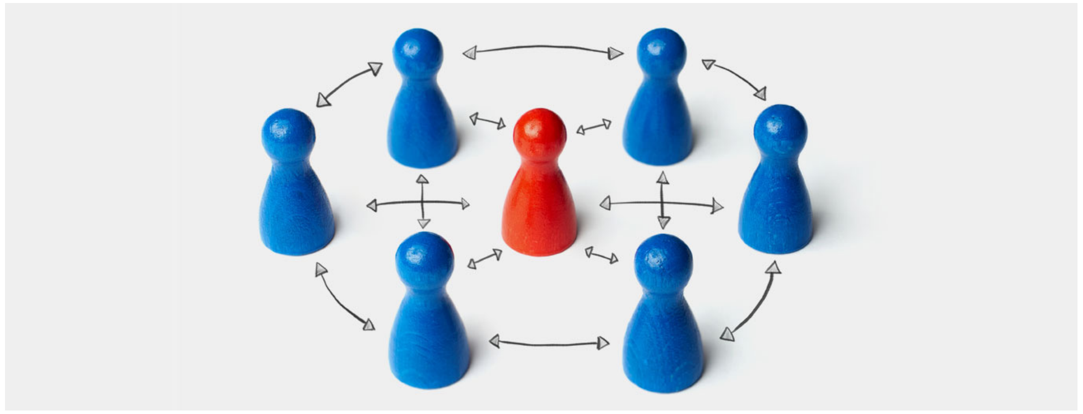
- The product owners guides the direction of the product by **ranking** the work based on its **business value.**
- The product owners work with their teams daily by **providing product feedback** and **setting direction on the next piece of functionality** to be developed/delivered.
- The product owners work with stakeholders, customers, and the teams to **define the product direction.**
- The product owners **create the backlog** for and with the team.
- The product owners decide **when and what to be released**.
- The product owners **accept or reject** work results.
- Product owner can be called as “Customer” in XP framework.



Product Owner
Product Leadership
• Vision
• Product strategy and roadmap
• Product backlog prioritisation
• Stakeholder management
User research
Sprint goal selection
Product backlog refinement
Close Collaboration
Design & Technology Leadership
• UX design
• Development incl. architecture and Q&A
• Self-organisation
Development Team
Scrum Master
Process Leadership
• Process and method coaching
• Collaboration facilitation
• Organisational change

Team facilitator

- The third role typically seen on agile teams is of a **team facilitator**, a servant leader.
- This role may be called a project manager, scrum master, project team lead, team coach, or team facilitator.
- The team facilitators **educate stakeholders** around why and how to be agile.
- The team facilitators **promote collaboration and conversation** within the team and between teams.
- The facilitators change or remove these organizational impediments to support delivery teams.
- The facilitators support the team through mentoring, encouragement, and support.



Questions

Who is accountable for the Product Backlog when it comes to content and order?

a. The Product Owner and the Developers

b. The Product Owner

c. The Scrum Master

d. The Stakeholders and the Product Owner

e. The Stakeholders

B. Only the Product Owner is accountable for managing the Product Backlog, even if they collaborate with the rest of the Scrum Team and the Stakeholders.

Questions

The four primary roles of a servant leader include:

a. Shielding team members from interruptions.
b. Resolving conflicts.
c. Determining which stories to include in an iteration.
d. Assigning tasks to the team members.

A. Shielding team members from interruptions. The four primary roles of a servant leader are shielding the team from interruptions, removing impediments to progress, communicating the project vision, and "carrying food and water:'

Questions

Who is responsible for creating potential releasable increment?

a. The Developers, but only if the Scrum Master is happy with the technical solution proposed.

b. The Developers.

c. The Scrum Master.

d. The Product Owner.

B. Developers are the people in the Scrum Team that are committed to creating any aspect of a usable Increment each Sprint.

Questions

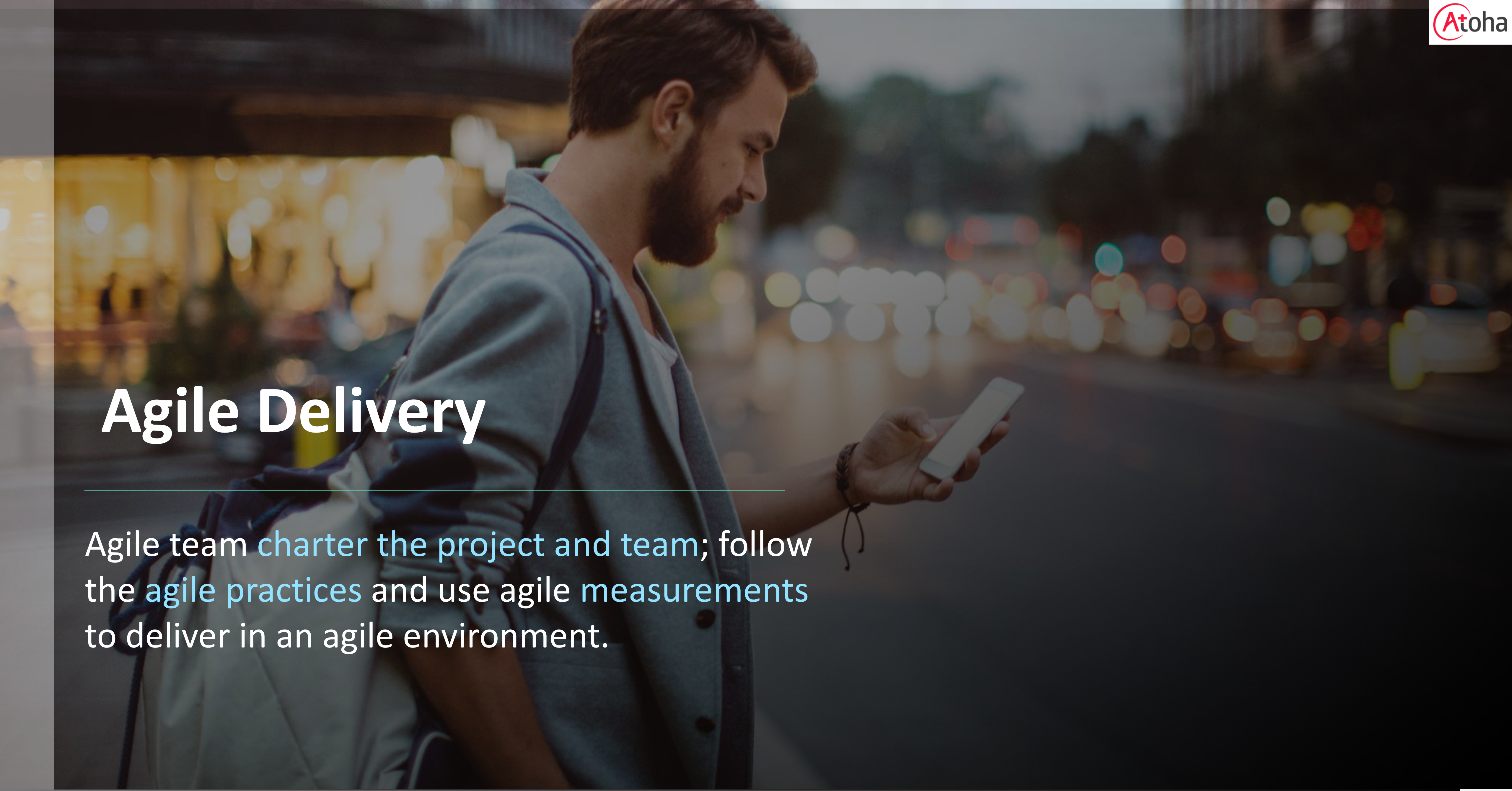
Scrum Teams are cross-functional. What does this mean?

a. Every Scrum Team member has all the skills required to create the Product. This is useful in case somebody is absent.

b. Scrum Teams can decide which skills they want to learn.

c. The Scrum Team, as a whole, has all the skills to create the Product.

C. Scrum Teams are cross-functional, meaning the members have all the skills necessary to create value each Sprint.



Agile Delivery

Agile team charter the project and team; follow the agile practices and use agile measurements to deliver in an agile environment.

Agile Project Charter

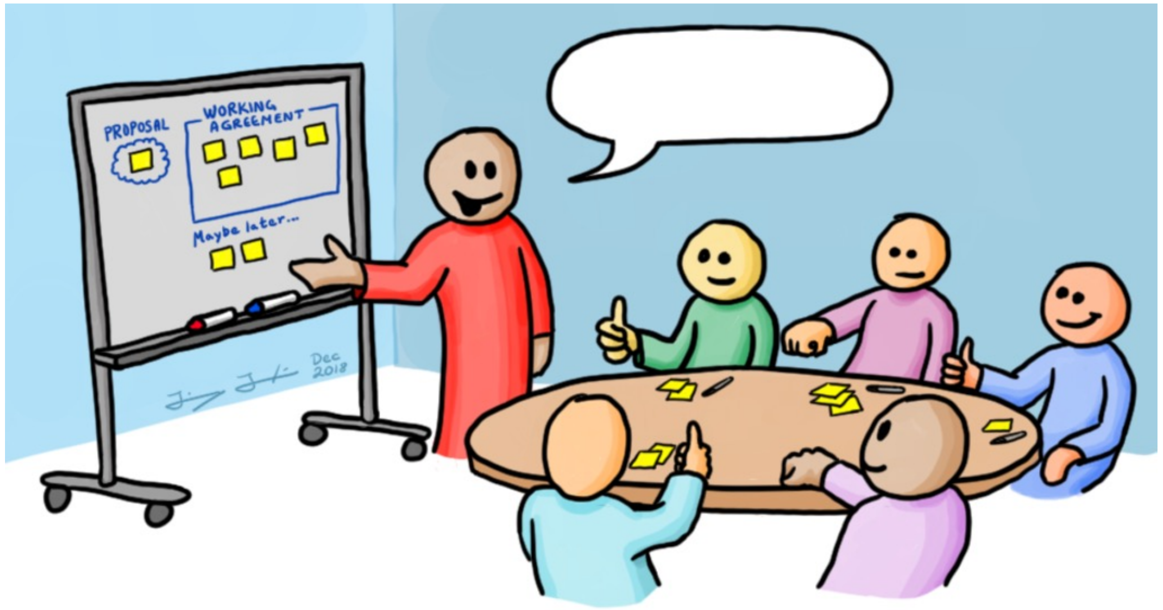
- Agile Charter Project needs the **project vision** or **purpose** and a clear set of **working agreements** at minimum.
- Chartering the project will be facilitated by the **servant leader**.

Why are we doing this project	PROJECT VISION
Who benefits and how?	PROJECT PURPOSE
What does done mean for the project?	RELEASE CRITERIA
How are we going work together	INTENDED FLOW OF WORK

Team Charter

A document that enables the team to establish its values, agreements, and practices as it performs its work together. Includes:

- ✓ Shared values: such as sustainable pace and core hours;
- ✓ Guidelines for communications and use of tools
- ✓ Decision-making guidelines
- ✓ Conflict resolution measures
- ✓ Meeting time, frequency, and channel
- ✓ Other team agreements e.g. shared hours, improvement activities



Agile Events

Backlog Refinement	Iteration Planning Meeting	Daily Standup
• A meeting with no more than 1 hour per week for refining stories for the next batch of work. • It can be called as grooming the backlog. • Product owner works with the team to prepare some stories for the upcoming iteration during one or more sessions in the middle of the iteration. • All the team participate in the meeting.	• A timeboxed meeting no longer than 8 hours. • Participants: the delivery team, the product owner, and possibly other stakeholders or SMEs as needed. • The meeting requires a prioritized backlog and a goal for iteration. • The team discuss how the work will be completed • Teams estimate what can be completed, but cannot predict with 100% what can be delivered.	• A timeboxed meeting no longer than 15 minutes. • It can be called as standup meeting. • In flow based Agile, focus is on team's throughput. • Meeting is for realizing the problem – not solving the problem. • The meeting is for development team only.

Agile Events

Demonstrations/Reviews	Retrospectives
• A timeboxed meeting with no more than 4 hours. • The team demonstrates all completed work items at the end of iteration in iteration-based agile. • In flow-based agile, the team demonstrates completed work when it is time to do so. • Product owner sees the demonstrations and accepts or declined stories. • The recommendation frequency for the demo meeting is at least once every 2 weeks. • The meeting is to getting feedback early and learn fast to frequently deliver working product.	• Opportunity to learn about, improve, and adapt its process, people, interaction and tools. • Holds retrospective at the end of iteration; or team can decide retrospect when: o completes a release or ships something. o more than a few weeks since the last retrospective o the team appears to be stuck o the team reaches any other milestone • Team facilitator facilitates the event. • The meeting output are list of ranking action items. • The appropriate number of work will be chosen to work on for the next iteration (or added to the flow if flow-based)

Execution practices

1. Continuous Integration
   - Frequent incorporate works into the whole.
   - Retest to determine that the entire product still works as intended.
2. Test at all levels
   - Employ system-level testing for end-to-end information and unit testing for the building blocks.
   - Check for the need of integration testing and where.
3. Acceptance Test-Driven Development (ATDD).
   - Moves the testing focus from the code to the business requirement.
   - Defines the acceptance criteria from user’s perspective and creates the test before writing code.

Execution practices

4. Test-Driven Development (TDD)

- Test is written before the code is written (test-first).
- Developers begin a cycle of writing code and running the tests until the code passes all tests (refactor).

5. Behavior Driven Development

- An extension to TDD where team starts by writing a system behaviors, then automated test scripts.
- Team code, retest and refactor to ensure that the entire product works as required behaviors.

6. Spikes

- A timeboxed research or experiments.
- Architectural spike used for research and risk-based spike used for risk mitigation.

Agile Project Challenges

Unclear purpose or mission

Agile chartering for purpose—vision, mission, and mission tests

Unclear working agreements

Agile chartering for alignment—values, principles, and working agreement

Unclear team context

Agile chartering for context—boundaries, committed assets, and prospective analysis

Unclear requirements

Craft a product vision; clarify the expectations and value of a requirement. Progressively decompose roadmap into backlog of smaller, concrete requirements

Agile Project Challenges

Poor user experience

User experience design practices included in the development team involve users early and often.

Inaccurate estimation

Reduce story size by **splitting stories**. Use **relative estimation** with the **entire team** to estimate. Consider agile **modeling** or **spiking**.

Unclear work assignments/work progress

Team **self-manage** their work. Consider Kanban boards to see the **flow of work**. Consider a **daily standup** to walk the board and see what work is where.

Team struggles with obstacles

A **servant leader** help clear these obstacles; or **escalate obstacles** (or roadblocks) the team or servant leader has not been able to remove.

Agile Project Challenges

Work delays due to insufficiently refined PBIs

Product owner and team workshop stories together. Create a **definition of ready** for the stories. Consider **splitting stories** to use smaller stories.

Defects

Consider the technical practices: **pair work**, **collective product ownership**, **pervasive testing** (test-driven and automated testing approaches) and a robust **definition of done**

Work is not complete

Team define definition of done for stories including **acceptance criteria**. Also **add release criteria** for projects.

Technical debt

Refactoring, **agile modeling**, **pervasive testing**, **automated code quality analysis**, **definition of done**.

Agile Project Challenges

Atoha

Too much product complexity

Apply the agile principle of "**Simplicity**--the art of maximizing the amount of work not done" to reduce complexity.

Slow/no improvement in the teamwork process

Capture **no more than three items** to improve at each retrospective. Ask the servant leader to help the team learn how to integrate those items.

Too much upfront work leading to rework

Create spikes to learn; **measure the WIP**; **shorten iterations** and create a robust **definition of done**.

False starts, wasted efforts

Ask the product owner to become an **integral part of the team.**

Agile Project Challenges

Inefficiently ordered PBIs

Rank with value including cost of delay divided by duration (CD3) and other value models

Rush/wait uneven flow of work

Plan to the **team's capacity**; **stop multitasking** and be **dedicated** to one team.

Work as pairs, a swarm, or mob to even out the capabilities across the entire team.

Impossible stakeholder demands

Servant leadership to work with this stakeholder (and possibly product owner).

Unexpected or unforeseen delays

Use **Kanban boards** to see the flow of work and WPI limits; **track impediments** and impediment removal on an impediment board.

Siloed -team

- **Self-organize** as **cross-functional** teams. Use servant leadership skills to help the managers understand why agile needs cross-functional teams.

Measurements in Agile

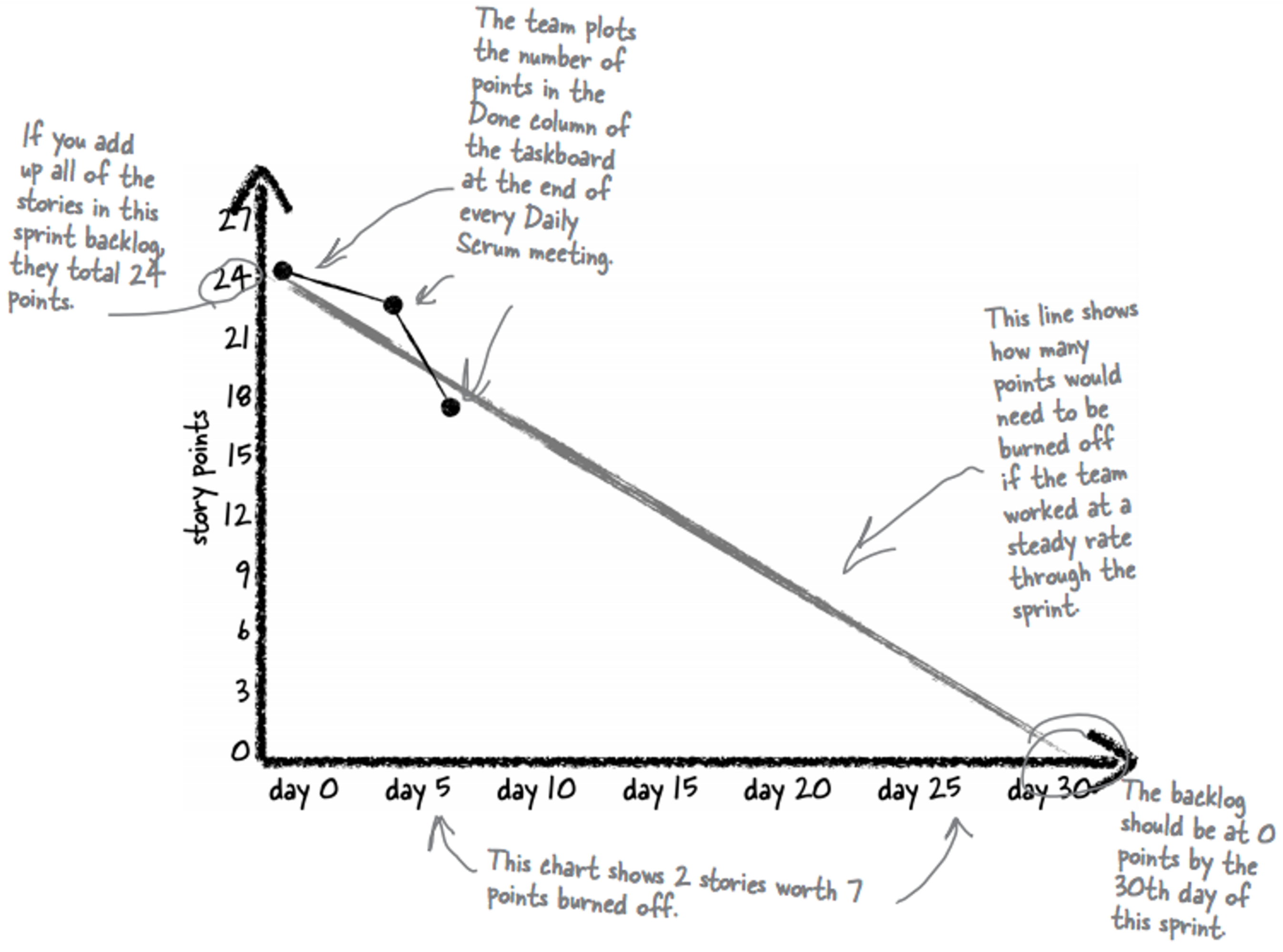
- Agile measures **what the team delivers**, not what the team predicts it will deliver.
- If there is low variability in the team's work and if the team members are not multitasking, the team's capacity can become stable which allows better prediction for the next couple of weeks.
- Agile teams use empirical data and replans further small increments to manage the project uncertainty.
- Iteration-based Agile can use **burnup, burndown** charts to see where is the project going overtime and velocity for measurement.
- Flow-based Agile can uses **lead time**, **cycle time** for measurements

Case: Sponsors usually want to know when the project will be done. The team established a reliable velocity (average 50 story points per iteration) or the average cycle time; there are about another 500 points remaining.

Then the team estimates it has about 10 iterations remaining, approximately 5 months if an iteration spans two weeks.

Burndown chart

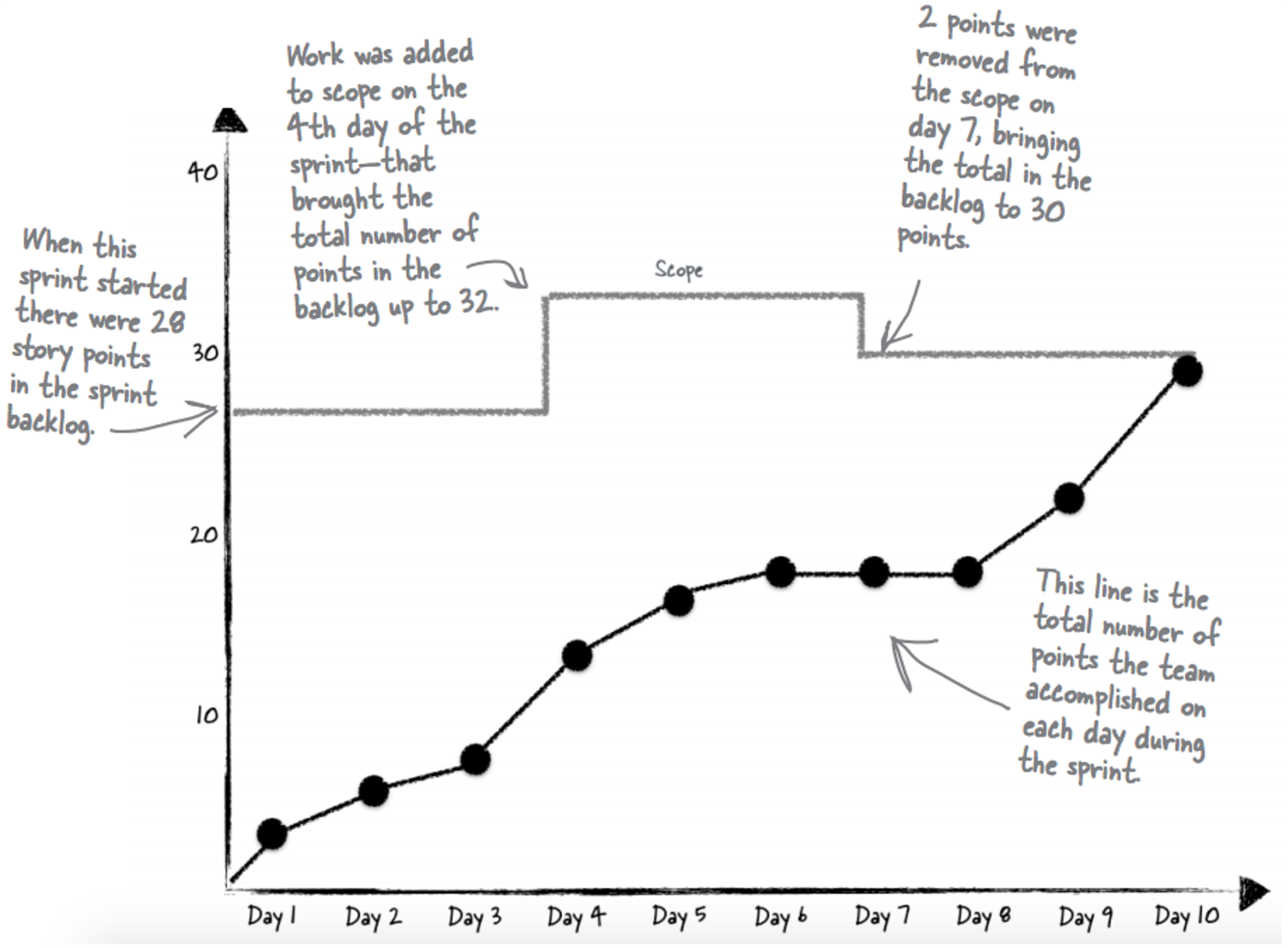
- Once a team has assigned a story point value to all of the user stories in the sprint backlog, they can use burndown charts to get a handle on **how the project is progressing**.
- A burndown chart is a **simple line** chart that shows **the remaining story points**.
- Using a burndown chart, it's clear to everyone on the team **how close they are to achieving their sprint goals**.
- NOT related to the project costs or the business value
- NOT related to the productivity of the team or to the individual team member.



Burndown charts and velocity help the whole team stay in control of the sprint.

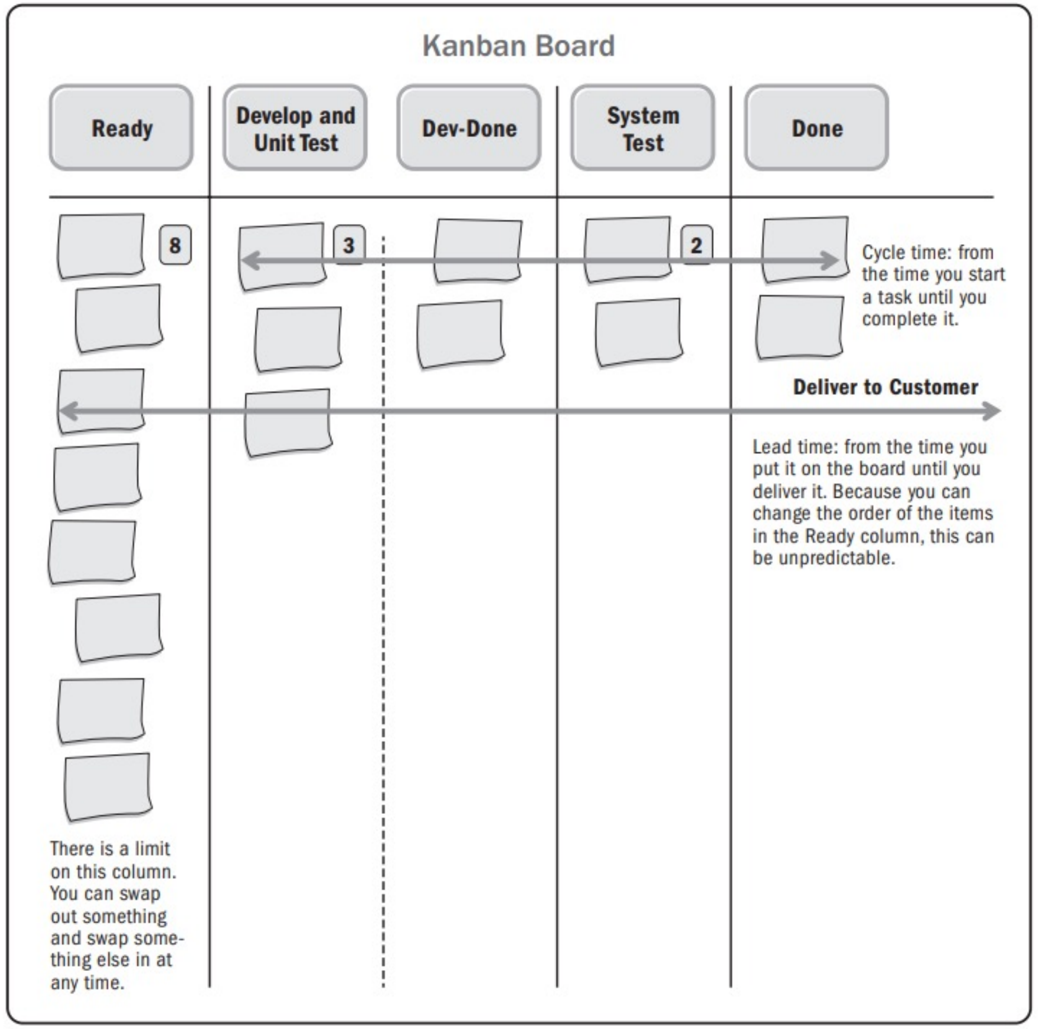
Burnup chart

- Burn-up chart show **the work completed**.
- When stories are added or deleted from the scope it's obvious by looking at the **scope line.**
- Because the scope is tracked on a different line from the number of points accomplished, **it's clearer when the scope is changing**.
- Whether burndown or burnup charts is used, at the end of the iteration, team might base their next measure of **capacity** on what they completed in this iteration.
- **Velocity**, the sum of the story point sizes for the features actually completed in this iteration, allows the team to plan its next capacity more accurately by looking at its historical performance.



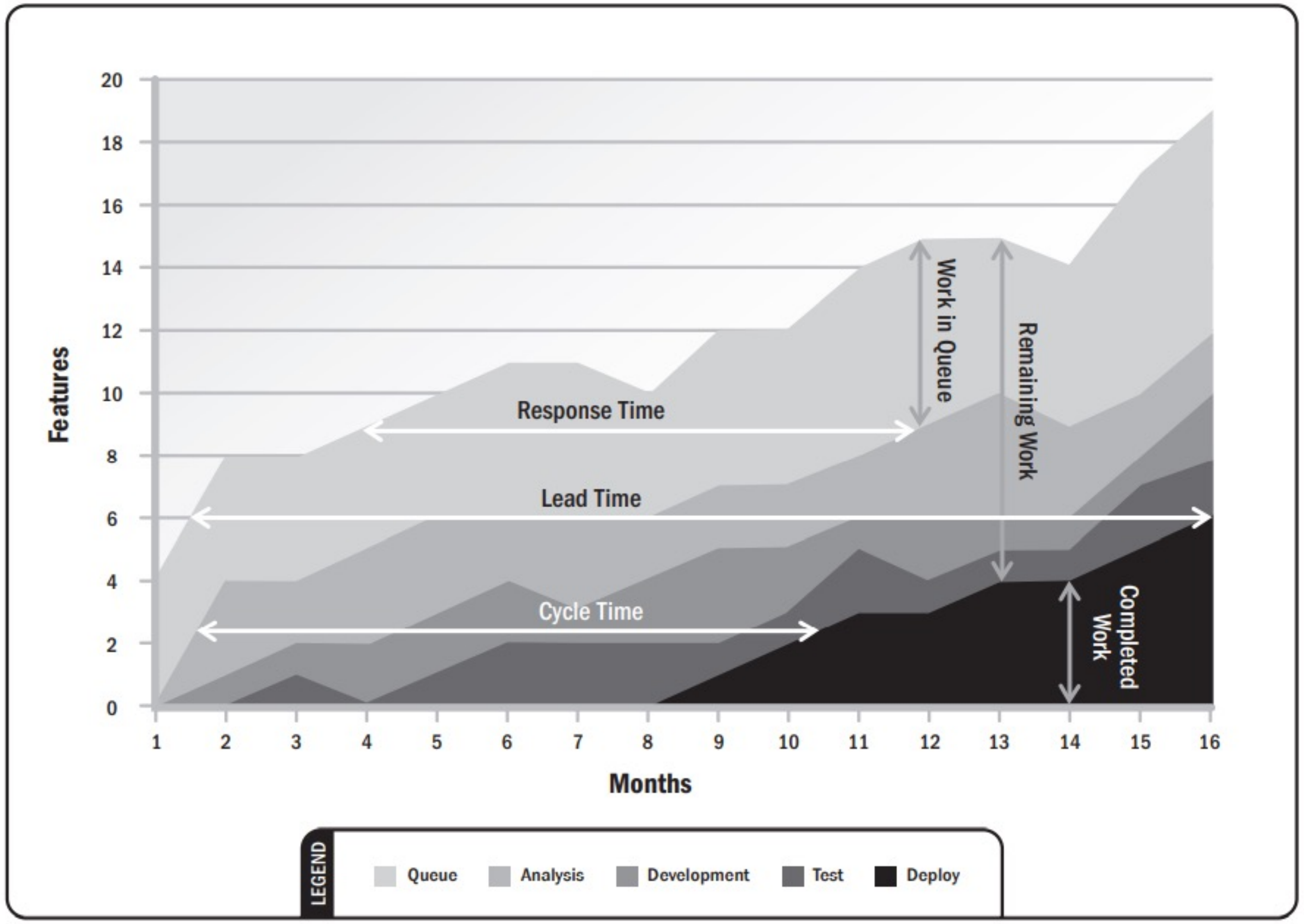
Kanban Board

- **Lead time:** the total time it takes to **deliver an item**, measured from the time it is added to the board to the moment it is completed;
- **Cycle time**: the time required to **process an item and response time** (the time that an item waits until work starts). Teams measure cycle time to see bottlenecks and delays.
- The **work in progress (WIP)** limits at the top of each column, allows the team to see how to pull work across the board.
- When the team has met its WIP limits, the team cannot pull work from the left into the next column.
- Each feature is unique, so its **cycle time is unique**.
- Smaller features have smaller cycle times. The product owner wants to see throughput, so the product owner creates smaller features or works with the team to do so.


Kanban Board
Ready
Develop and Unit Test
Dev-Done
System Test
Done
8
3
2
Cycle time: from the time you start a task until you complete it.
Deliver to Customer
Lead time: from the time you put it on the board until you deliver it. Because you can change the order of the items in the Ready column, this can be unpredictable.
There is a limit on this column. You can swap out something and swap some-thing else in at any time.

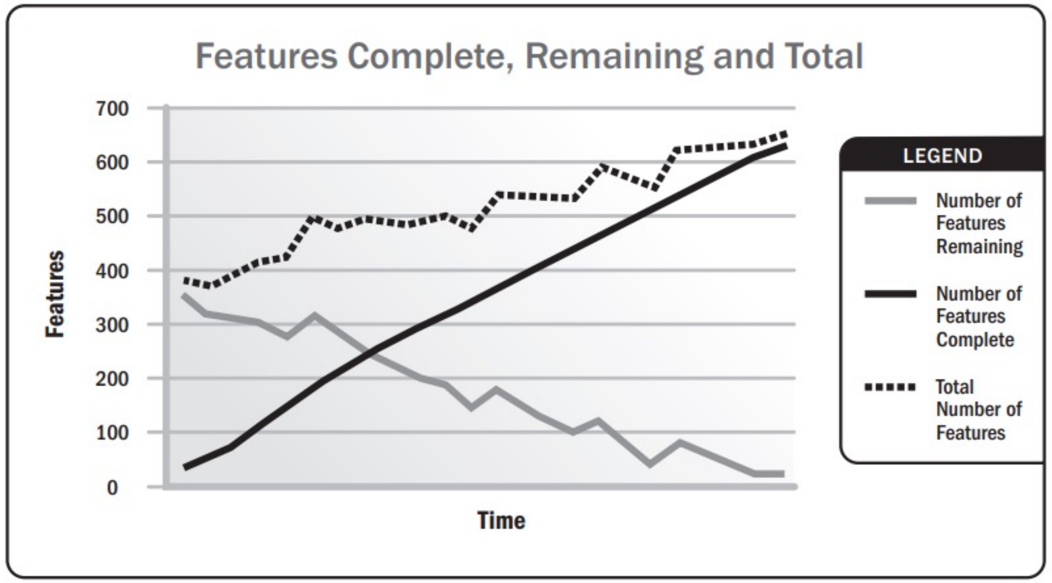
Cumulative Diagram

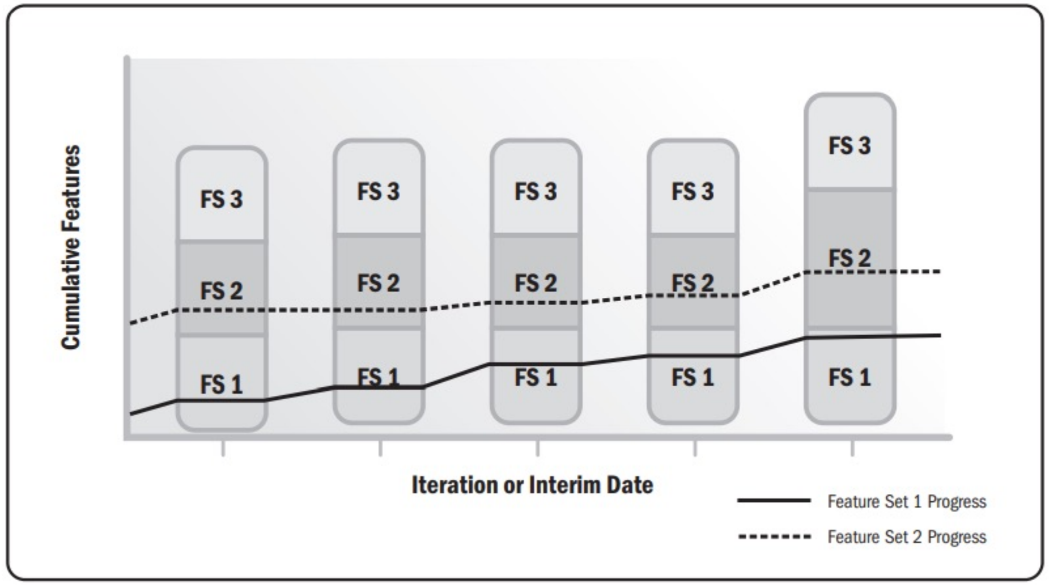
- A cumulative flow diagram shows the **work in progress across a board**.
- If a team has many stories waiting for test, the testing band will swell. Work accumulation can be seen at a glance.
- Teams have trouble with accumulating work: the team has **work in progress** instead of work completed.
- When teams have a lot of work in progress, they delay their overall feature delivery.
- Teams can **limit their work in progress** to remove bottle neck.



Feature Chart & Product Backlog Burnup Chart

- When teams measure only **story points**, they measure **capacity, not finished work**.
- Team can measure **completed work** in a **feature burnup/burndown chart** and in a product backlog burnup chart.
- The features complete line shows that the team completes features at a regular pace. The total features line shows how the project's total features changed over time.
- The features remaining burndown line shows that the rate of feature completion varies. The burndown line changes when features are added to the project.
- The team can show its **completed value** with a **product backlog burnup** in case the entire feature can not completed until several more time periods have passed.





Earned Value Calculation in Agile

- Burn-up chart can be used to measure Earn Value.
- **SPI** = Completed Features/Planned Features
- **CPI** = Earned Value/Actual Costs

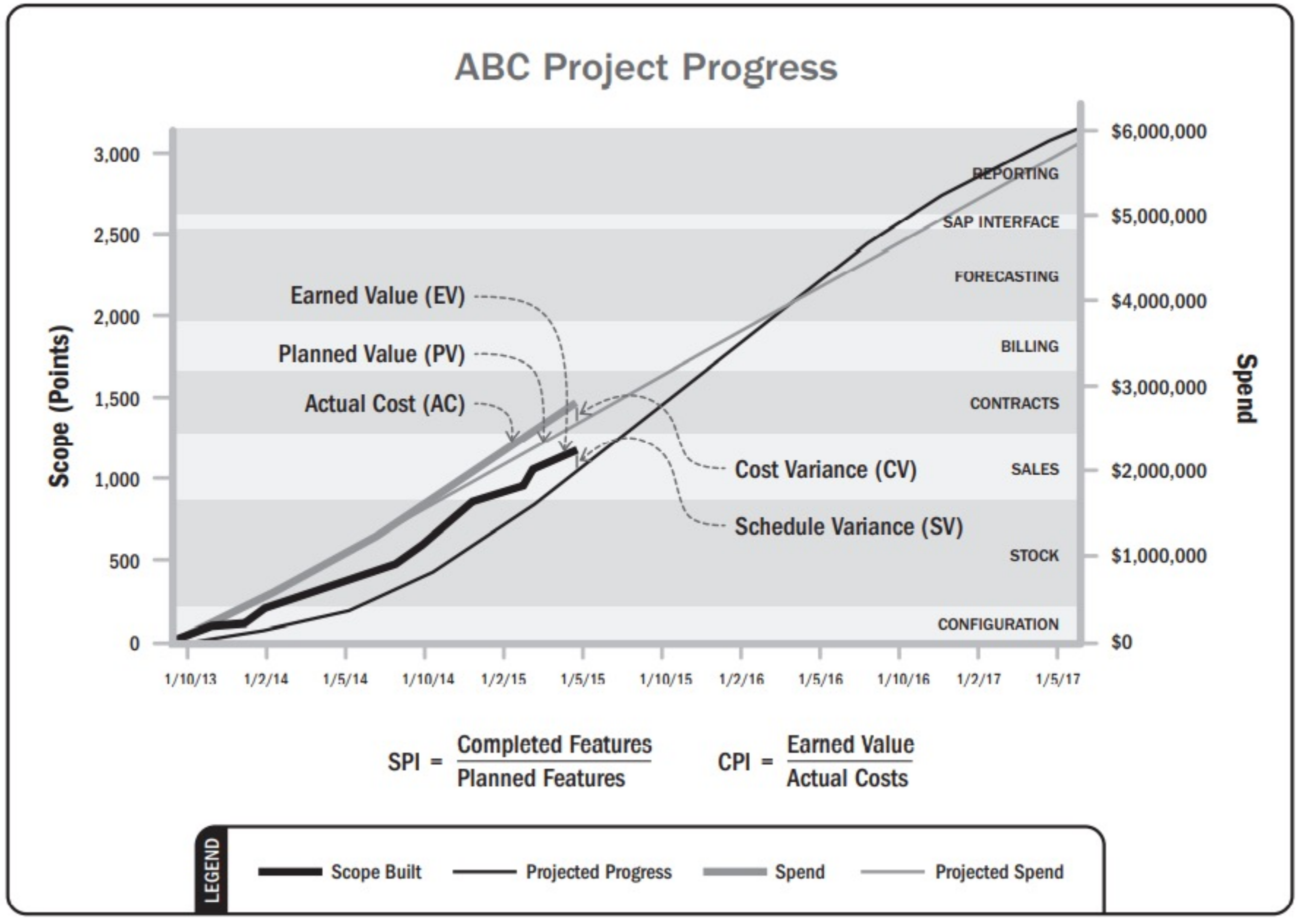
Case:

If the team planned to complete 30 story points in an iteration, but only completed 25 then:

SPI = 25/30 or 0.83 (the team is working at only 83% of the rate planned).

CPI = $2.2M / $2.8M = 0.79 (according to the figure)

This means a result of only 79 cents on the dollar compared to plan.



Questions

Your sponsor has asked for clarification on when releases of your product will ship and what those releases will contain. Which agile deliverable would best address this need?

a. Product demo.

b. Product roadmap

c. Product backlog

d. Product owner.

B. The product roadmap shows release dates and the high-level contents of releases, so it would be the best deliverable for answering these questions.

Questions

What is velocity not used for? Select one.

a. Gauging the team's work capacity.

b. Checking the validity of the release plan.

c. Getting a sense of the amount of work done per iteration.

d. Defining feature requirements

D. Velocity is a versatile metric that does indeed provide insight into team capacity, release plan validity, and finally the amount of work done per iteration.

Questions

A short timeboxed period set aside to investigate and eliminate a risk is called a. Select one.

a. Risk stop.
b. Spike of iteration.
c. Risk-based spike.
d. Risk-first design.

C. A short timeboxed period set aside to investigate and eliminate a risk is known as a risk based spike. The other options are made-up terms.

Questions

The three questions that are answered in daily stand-up meetings aim to:

Select one:

a. Identify problems and describe accomplishments.

b. Identify opportunities and celebrate accomplishments.

c. Fix problems and share accomplishments.

d. Fix problems and agree upon work planned

A. The three questions asked at stand-up meetings are: "What have you worked on since the last meeting?" "What do you plan to finish today?" and ''Are there any problems or impediments to your progress?" These questions describe accomplishments and work planned and identify problems. Identifying opportunities, celebrating accomplishments, and fixing problems are worthy efforts, but they are not done in the stand-up meeting.

Questions

On agile projects, the term "cycle time" usually refers to:

Select one:

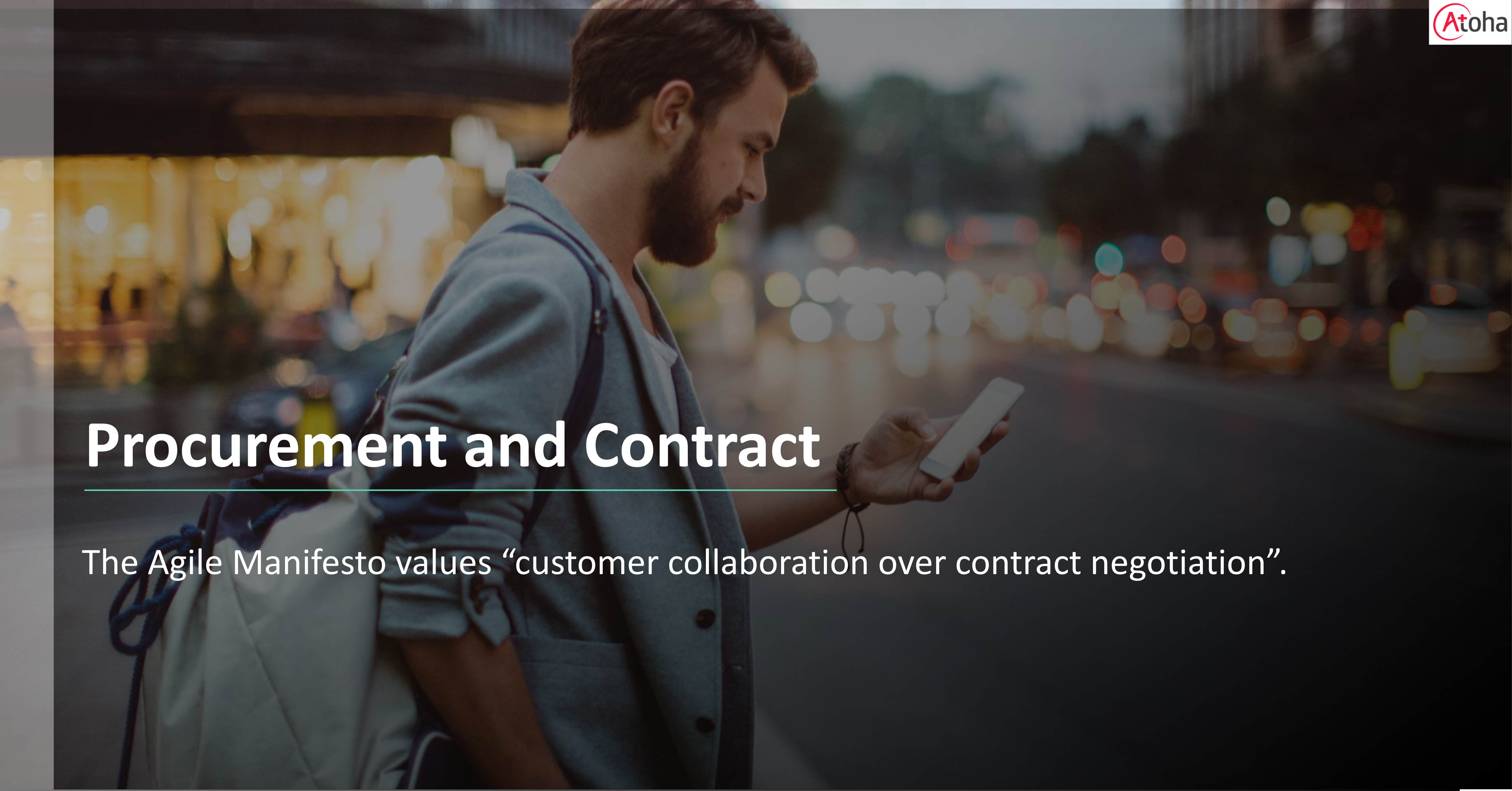
a. The average duration of an iteration, based on the team's capacity

b. The time required for the team to complete a work item, from start to finish

c. The time required for subject matter experts to review the product

d. The time periods ("cycles") between the releases of a product

B. On agile projects, cycle time is typically used to refer to the time it takes the team to complete the development of a work item, from start to finish. Officially, this concept is a subset of lead time, measuring how long it takes for something to go through part of a process.



Procurement and Contract

The Agile Manifesto values “customer collaboration over contract negotiation”.

Contracting in Agile Project

- An agile approach **requires more trust** between the parties than the traditional approach.
- The agile approach also requires the customer to be more **involved in providing feedback** on iteration deliverables, **reprioritizing the backlog**, and **ranking the value of change requests** against the remaining work items.
- For trusting, invested clients, agile contracts are great tools for **extracting more value**, and they give those clients a competitive advantage.
- For untrusting or hands-off clients, agile contracts will be a **tough sell** and may not be suitable.
- Agile values a collaborative approach which pursues a **shared-risk-reward relationship**.

Contracting techniques

Contracting techniques	Description
Multi-tiered structure	• Different aspects are described in different documents • Fixed items (e.g., warranties, arbitration) can be locked in a master agreement. • Items subject to change in a schedule of services. • More dynamic items such as scope, schedule, and budget can be formalized in a lightweight SOW.
Emphasize value delivered	• Milestones and payment terms can be structured based on value-driven deliverables in order to enhance the project's agility
Fixed-price increments	• Decompose the scope into fixed-price micro-deliverables, such as user stories. • For the customer, more control over how the money is spent. For the supplier, limits the financial risk of over-commitment to a single feature or deliverable
Not-to-exceed time and materials.	• Limit the overall budget to a fixed amount. • Customer are allowed to incorporate new ideas and innovations into the project not originally planned. • Contingency hours could be planned into the maximum budget if considered helpful.

Contracting techniques

Contracting techniques	Description
Graduated time and materials	• Financial risk can be shared. • A higher hourly rate can be rewarded when delivery is earlier than the deadline. • Rate reduction can be suffered for late delivery
Early cancellation option	• When an agile supplier delivers sufficient value with only half of the scope completed, the customer should not be bound to pay the remaining if they no longer needs it. • The customer limits budget exposure and the supplier earns positive revenue for services no longer required.
Dynamic scope option	• Budget is fixed, risk of over commitment from supplier is limited • Supplier may offer the customer the option to vary the project scope at specified points.
Team augmentation	• Embed the supplier's services directly into the customer organization. • Funding teams instead of a specific scope preserves the customer's strategic discretion on what work should actually be done.
Favor full-service suppliers	• Contracting with multi-suppliers. • Favor each supplier deliver full value to reduce dependency with other suppliers.

Questions

Which of the following is true about agile contracts?

Select one:

a. They only work when the specs are fully defined.

b. They only work for time and materials agreements.

c. They need to be able to accommodate changes.

d. They cannot easily accommodate changes.

C. The need to accommodate changes is a major reason agile contracts were developed, so that is the correct answer.

Questions

You have been asked to outline the basics of agile contracting for your steering committee. Which of the following statements best describes the recommended approach to contracting on agile projects?

Select one:

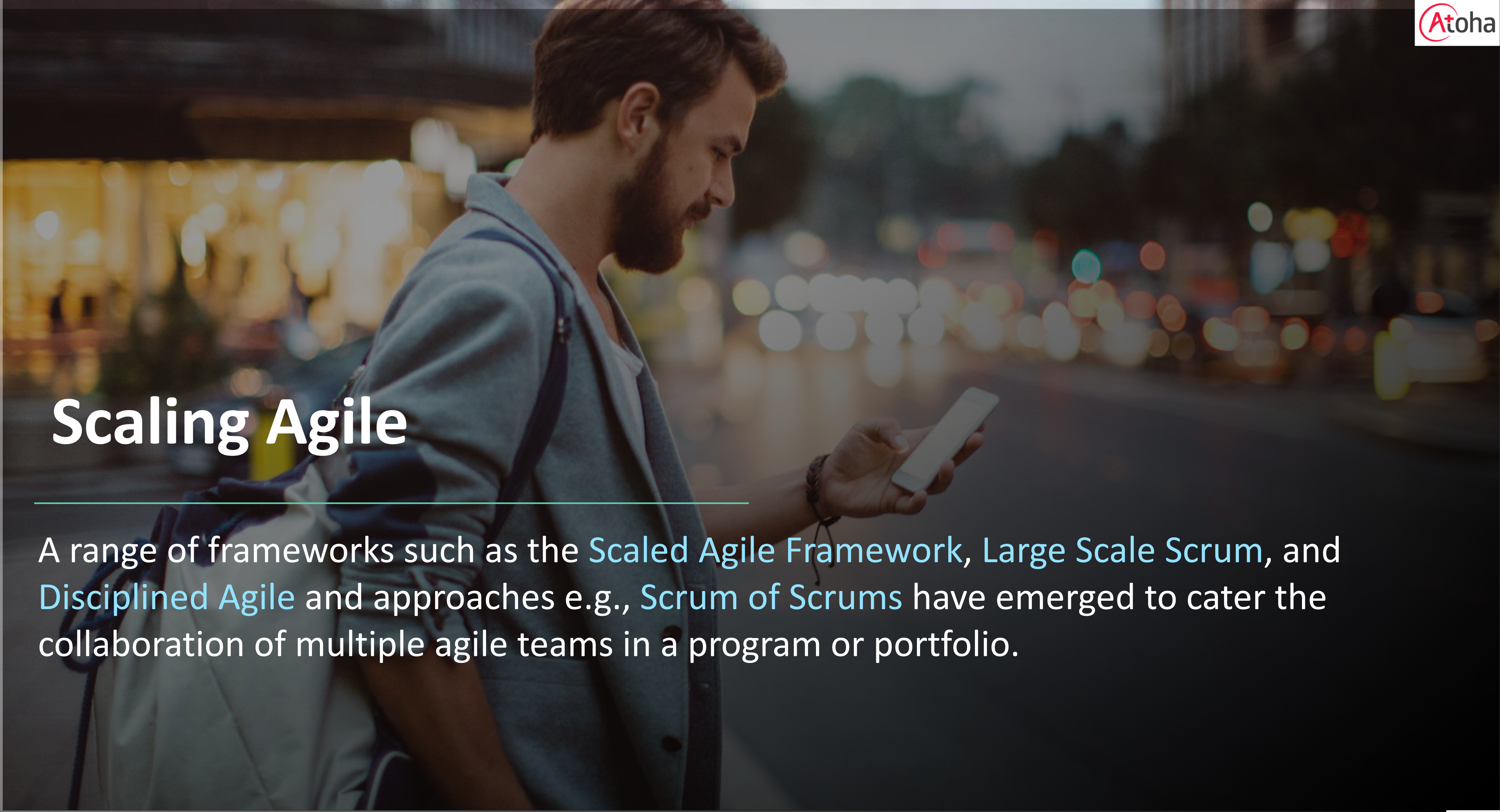
a. The contract is worded to allow for early completion of scope, and acceptance is based on items matching the original specification.

b. The contract is worded to allow for reprioritization of scope, and acceptance is based on items matching the original specification.

c. The contract is worded to allow for early completion of scope, and acceptance is based on items being fit for business purpose.

d. The contract is worded to allow for reprioritization of scope, and acceptance is based on items being fit for business purpose.

D. Two components common to agile contracts are an ability to reprioritize work and the goal of satisfying the business, rather than conforming to a spec.

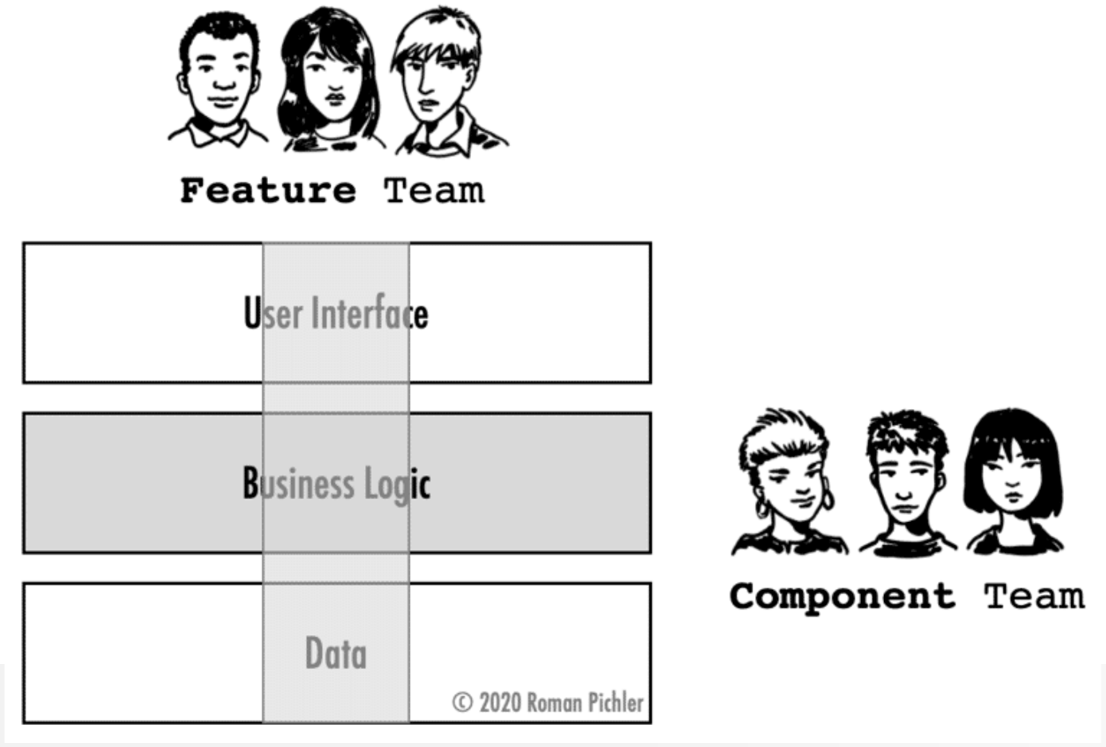


Scaling Agile

A range of frameworks such as the Scaled Agile Framework, Large Scale Scrum, and Disciplined Agile and approaches e.g., Scrum of Scrums have emerged to cater the collaboration of multiple agile teams in a program or portfolio.

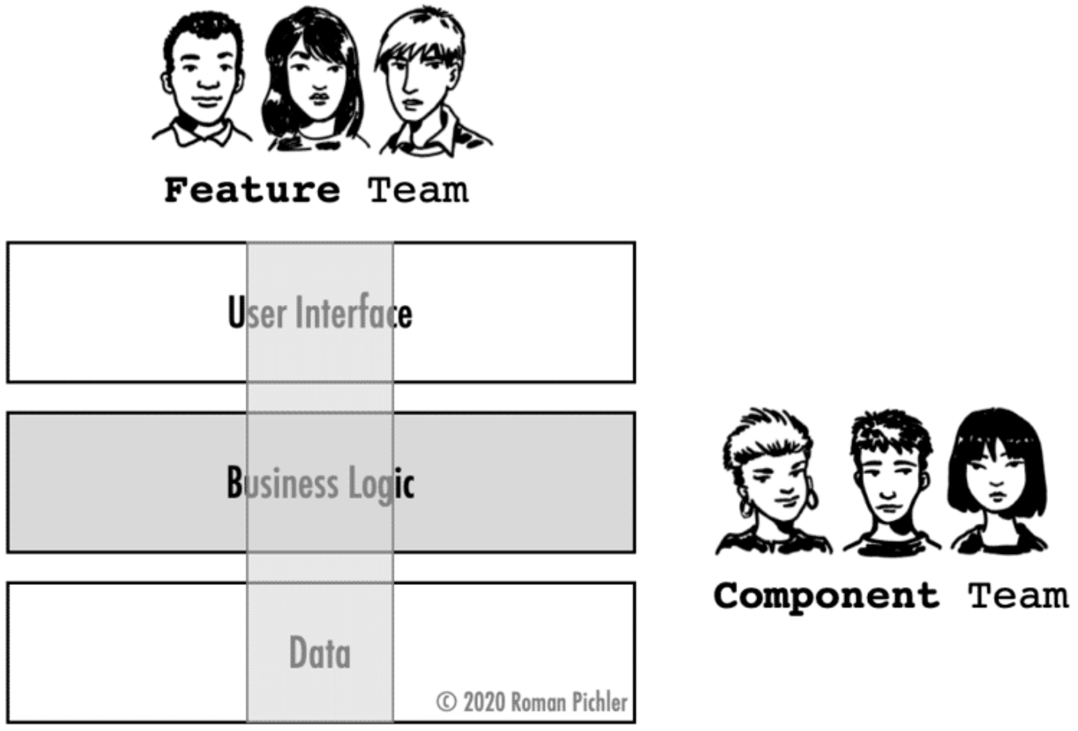
Feature teams vs. Component teams

- The architecture of the shop is currently divided into three separate layers.
  - There is the **user interface**, this is what you see and this makes it easier for the customers to interact with the shop, to see products and to order them.
  - Then there is the **business logic** layer which you don't see, but the application handles the business logic and connects the user interface with the database.
  - And finally, the **database** contains all the information regarding the products, the customers and orders, so everything that needs to be persisted.



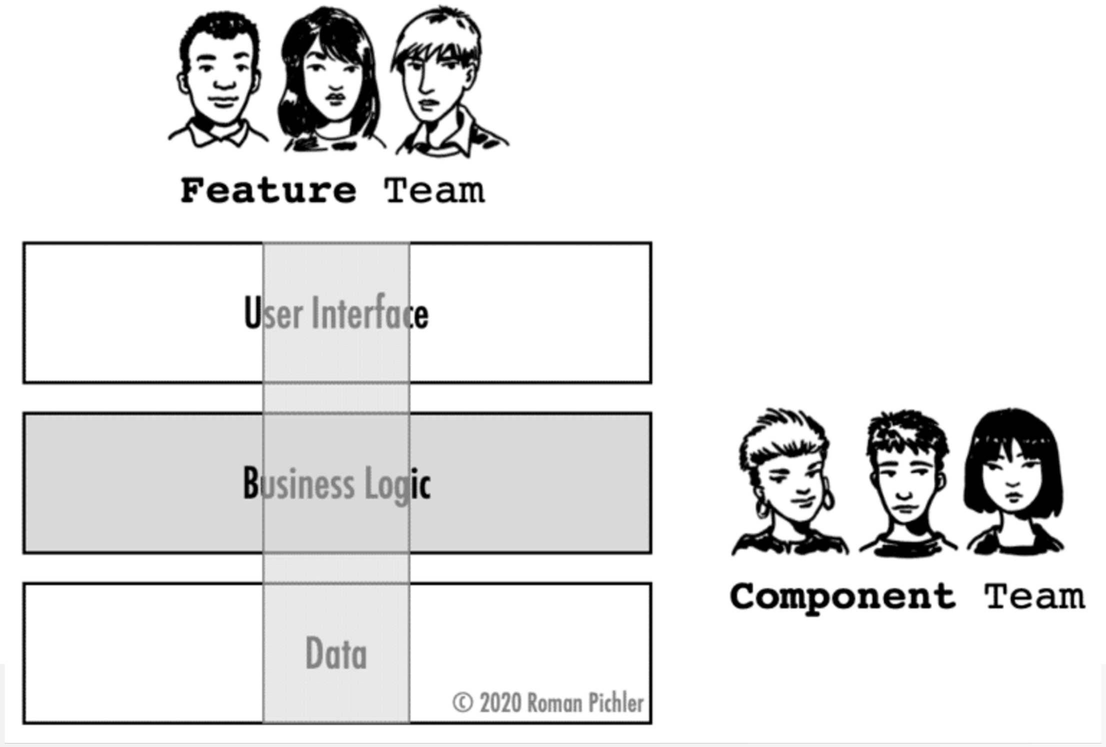
Feature teams

- For example, the feature is giving the customers the ability to order a Product. So a Product must be visible on the website and then when the customer clicks on, buy it, the application must handle the order in the database must store all this information for the processing or whatever the internal processes are. So you have to go through all the layers in order to implement most of the features.
- So a **Feature Team** will work through **all the layers** of the application to fulfill a customer or user need. A Feature Team is **cross-functional and cross-component** because it has all the skills needed to complete the feature and does that by working through all the layers or components of the application.



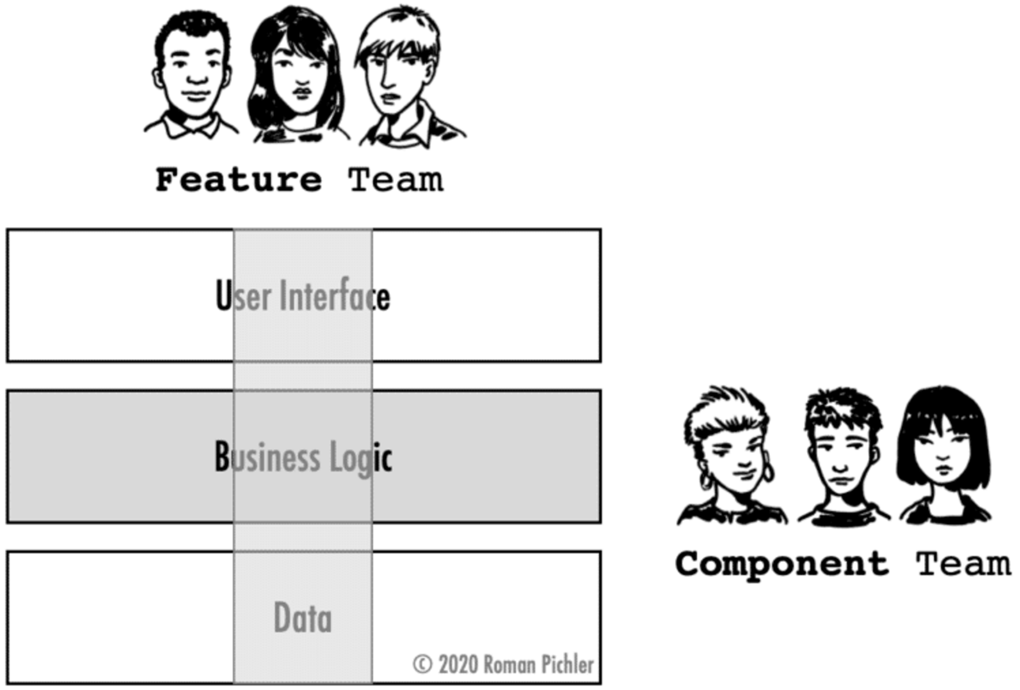
Component teams

- **Component Team**. Sometimes also referred to as the **Layer Team** is **focused on single or multiple components** of the system.
- A Component Team **alone will typically not be able** to deliver new functionality, that alone will fulfill a customer's need.
- For example, one Team would only handle the user interface, the UI, then that team alone would not be able to process orders, because it would not know how to work with a database layer or with the application layer and would have to rely on other teams to do their jobs.



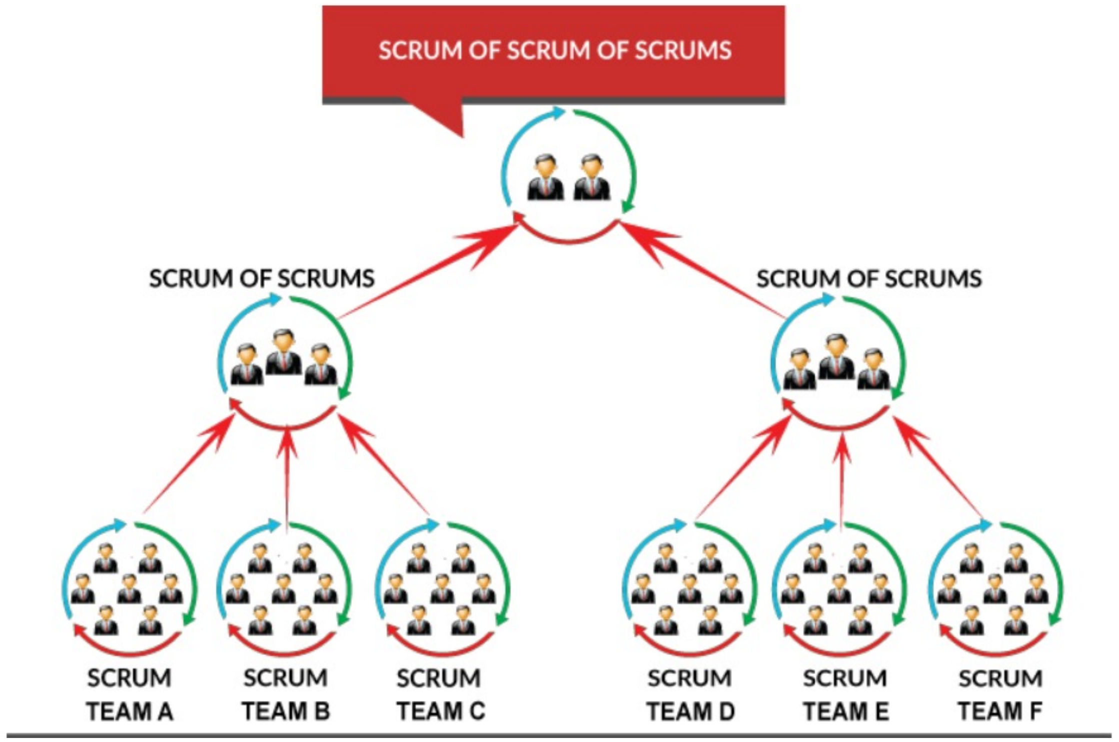
Feature teams vs. Component teams

- Component Teams **increases** the **dependencies** between the teams and reduces the chances of producing an **Integrated Product Increment** by the end of a Sprint.
- If one team for whatever reason, did not provide what they plan to by the end of the Sprint, then the other teams cannot work on their end. So that can cause problems. And this is why it says that typically the chances of producing an integrated Product Increment is lower when using Component Team.
- Feature Teams are desirable, as they are closer to the Cross-Functional aspect as it is described in the Scrum Guide. Nevertheless, Feature Teams are not mandatory in Scrum.



Scrum of Scrum

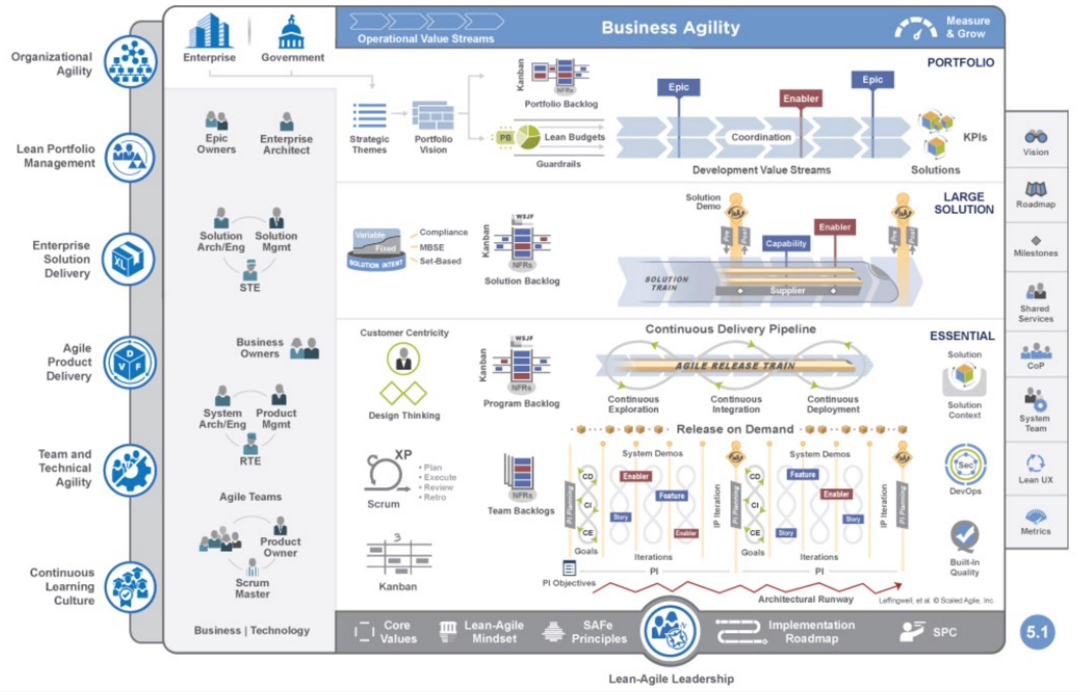
- When **two or more Scrum teams** consisting of three to nine members each need to coordinate their work instead of one large Scrum team, they use **Scrum of Scrum**.
- A representative from each team attends a meeting with the other team representative(s), potentially daily but typically two to three times a week.
- The goal is to ensure the teams are coordinating work and removing impediments to optimize the efficiency of all the teams.
- Large projects with several teams may result in conducting a **Scrum of Scrum of Scrums**.



Scaled Agile Framework (SAFe®)

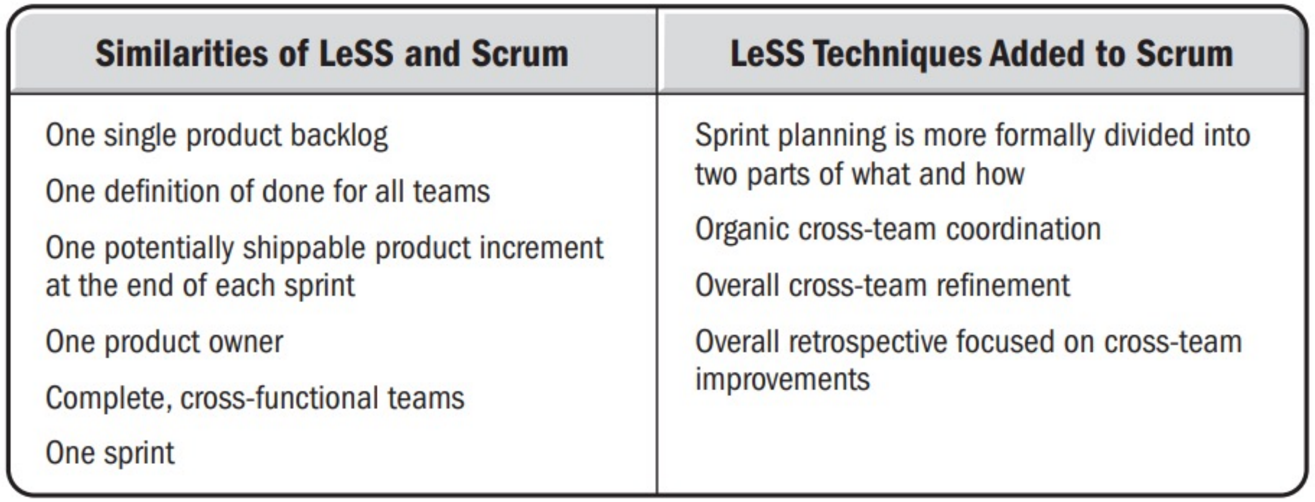
SAFe® focuses on **detailing practices**, **roles,** and **activities** at the portfolio, program, and team levels with an emphasis on **organizing the enterprise around value streams** that focus on providing continuous value to the customer. Safe® is focused on the following principles:

- Take an economic view.
- Apply systems thinking.
- Assume variability; preserve options.
- Build incrementally with fast, integrated learning cycles.
- Base milestones on objective evaluation of working systems
- Visualize and limit work in progress, reduce batch sizes, and manage queue lengths.
- Apply cadence; synchronize with cross-domain planning.
- Unlock the intrinsic motivation of knowledge workers.
- Decentralize decision making

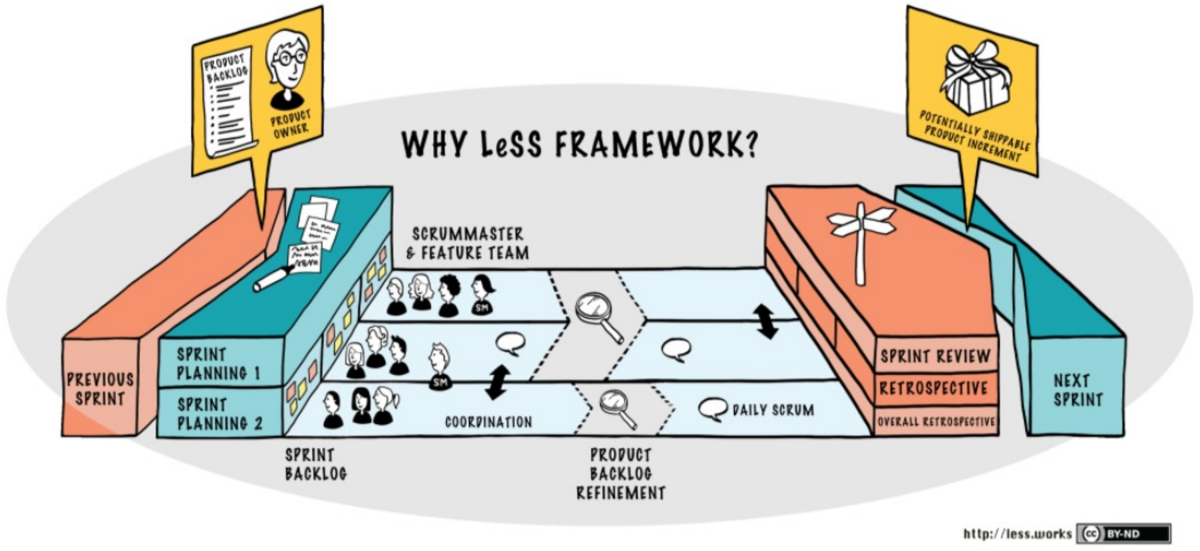


Large Scale Scrum (LeSS)

- Large Scale Scrum (LeSS) is a framework for **organizing several development teams** toward a common goal extending the Scrum method.
- The core organizing principle is to **retain as much as possible of the elements of the conventional single-team Scrum model**.
- This helps minimize any extensions to the model that might create unnecessary confusion or complexity.

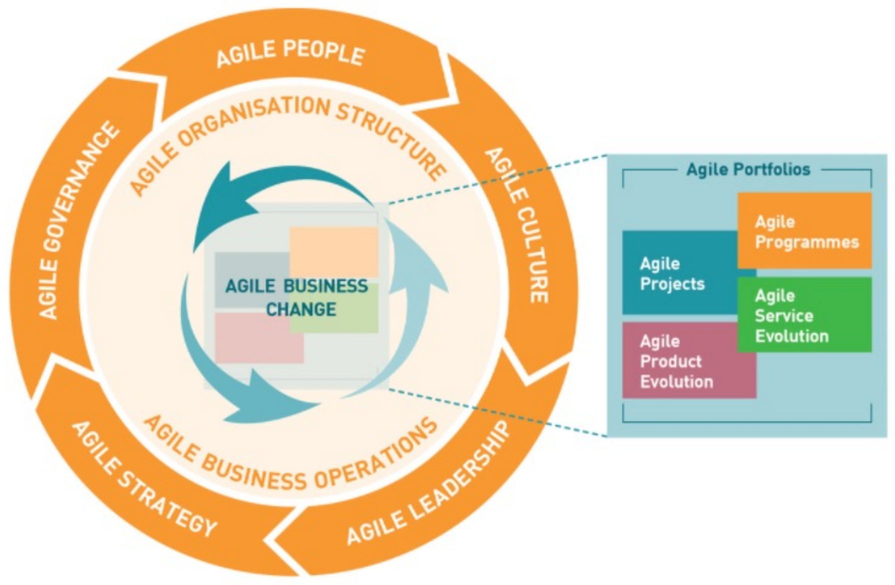


Similarities of LeSS and Scrum	LeSS Techniques Added to Scrum
One single product backlog	Sprint planning is more formally divided into two parts of what and how
One definition of done for all teams	Organic cross-team coordination
One potentially shippable product increment at the end of each sprint	Overall cross-team refinement
One product owner	Overall retrospective focused on cross-team improvements
Complete, cross-functional teams	
One sprint	



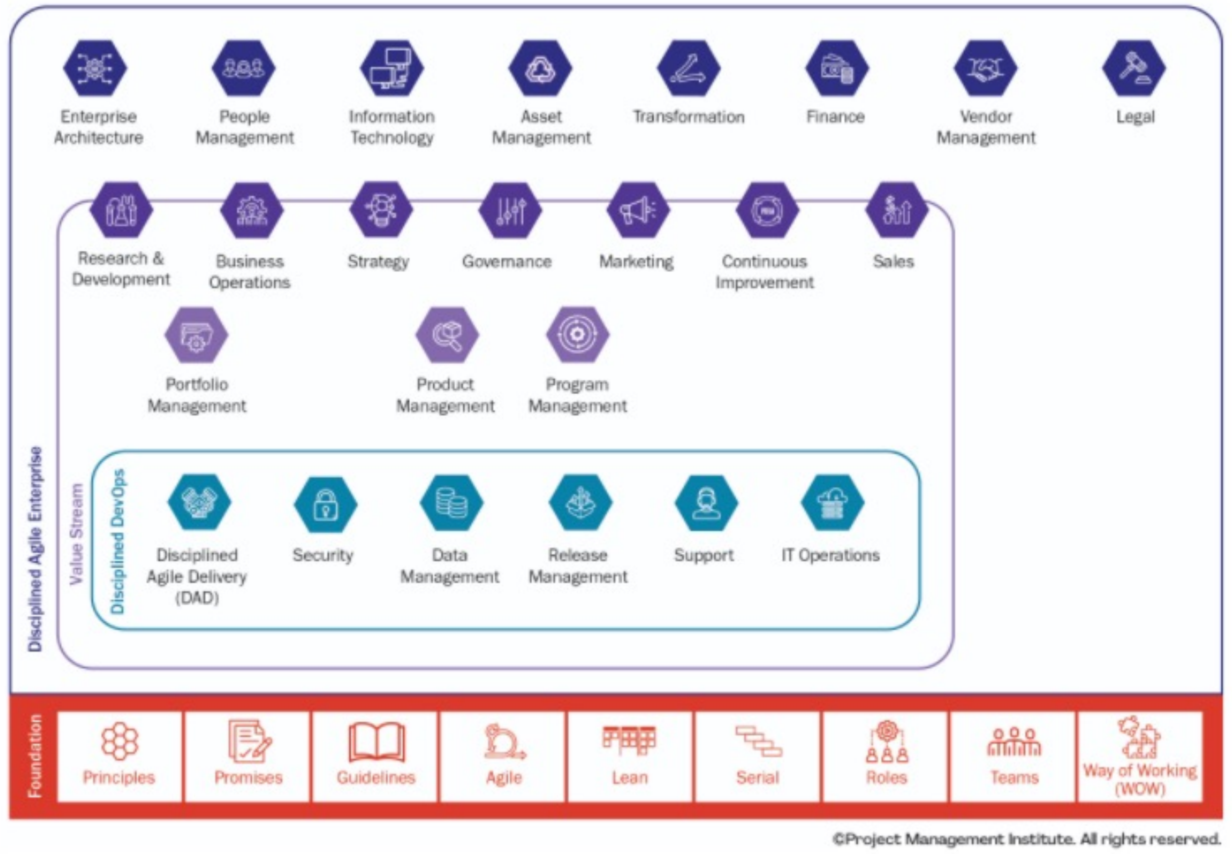
Enterprise Scrum

- Enterprise Scrum is a framework designed to **apply the Scrum method on a more holistic organizational level** rather than a single product development effort.
- The framework advises organization leaders to:
  - **Extend the use of Scrum** across all aspects of the organization;
  - **Generalize the Scrum techniques** to apply easily at those various aspects; and
  - **Scale the Scrum method** with supplemental techniques as necessary.



Disciplined Agile

- Disciplined Agile (DA) is a **process decision framework** that integrates several agile best practices into a comprehensive model. DA blends various agile techniques according to the following principles:
  - **People-first.** Enumerating roles and organization elements at various levels.
  - **Learning-oriented.** Encouraging collaborative improvement.
  - **Full delivery life cycle.** Promoting several fit-for-purpose life cycles.
  - **Goal-driven.** Tailoring processes to achieve specific outcomes.
  - **Enterprise awareness.** Offering guidance on cross-departmental governance.
  - **Scalable.** Covering multiple dimensions of program complexity.



Atoha

Thank You

Atoha Institute of Project Management

+84 28 6684 6687

cs@atoha.com