BASIC

AGILE & SCRUM FUNDAMENTALS

Your Path to Inspired Teams and Amazing Products.

TRAINING GUIDE

Continue your Agile learning journey with more Artisan Academy programs by going to our web site https://artisanagility.com/academy

For inspiration, motivation, and expert tips that will keep you being Agile, visit the Artisan Agility blog.

702.827.3500 • www.artisanagility.com

Table of Contents

Iconology

Icon	Definition
	A proposition that states the principles and underlying logic behind Artisan's approach. It is a context-neutral business truth that transcends boundaries of industry and geography. The briefcase is a reminder that insights are completely transferable to your organization.
	A common business practice that detracts from achieving exceptional leadership.
	Tips, gleaned from the insights, that can be used to build and sustain your agile culture.
	Personal reflections on the insights, ideas, and illustrations discussed in the course.
	Extended time to reflect and adapt the insights to your organization. You will discover and use tools for applying new ideas and potential solutions to your organization.

Introduction

Congratulations on taking a really important step on your way to joining the next generation of leaders by learning about Agile Development and the Scrum Framework. You are about to begin the first step on your training to become your company's creator of effective, high-performing product development teams. YOU are the one that will help ensure that your organization is set up to support the development team. YOU are the one who will ensure that the BEST product that can be built, WILL be built.

Scrum

Agile Software Development

Agile Development is a direct result of the increasingly complex world in which we live. In a very short period, computer power and computer usage has increased at an accelerating rate.

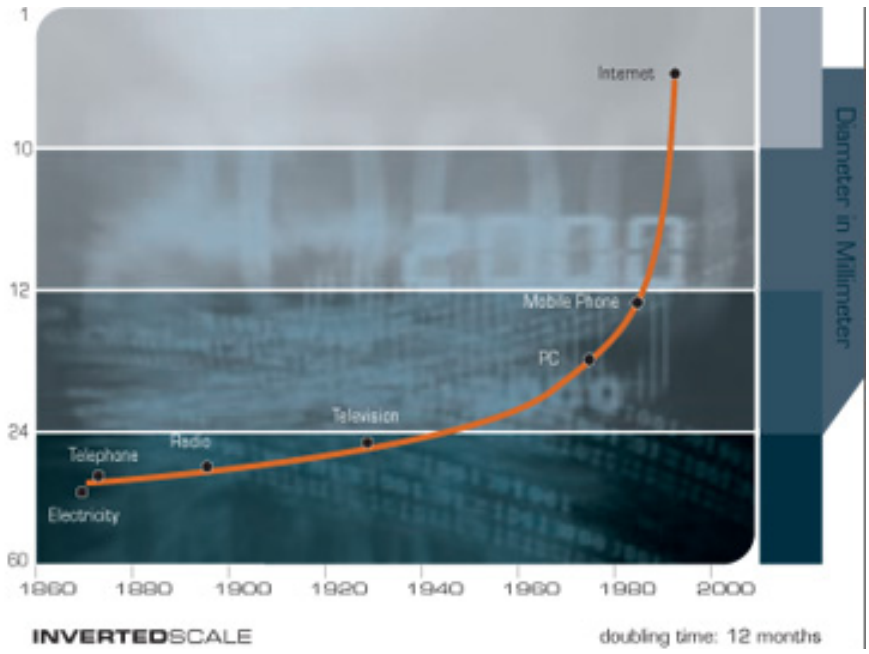


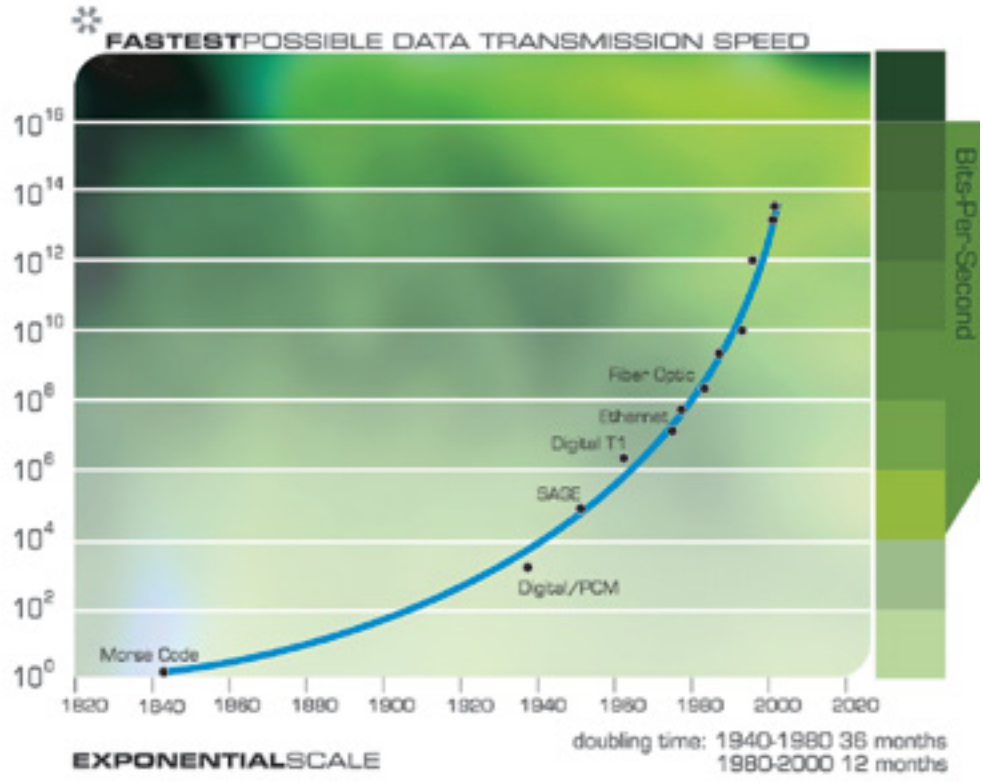
Figure 1: Mass Use of Inventions

Raymond Kurzweil is credit with coining the term "The Law of Accelerating Returns" when he noticed that computer processing power was not only increasing at an exponential rate, but <u>even the rate of increase was accelerating at an exponential rate!</u> Figure 1 and Table 1 show how long it took for ¼ of the population of the United States to use an invention. The faster an invention reaches mass use, the faster new technologies and paradigm shifts occur!

Table 1: Mass Use of Inventions Data Table

Year	Years	Invention
1873	46	Electricity
1876	35	Telephone
1897	31	Radio
1926	26	Television
1975	16	PC
1983	13	Mobile Phone

1991	7	Internet



Greater use of technology has required a significant improvement in data transmission bandwidth and speeds. From telegraph-based Morse Code to experimental technologies in 2016, data transmission has increased from a few characters a second to 1 terabyte.

That's 1,000,000,000,000 bytes!

According to Table 1, it took only 7 years to get ¼ of the US population using the internet. Table 2 shows the explosive growth of internet hosts (servers) to support the users of the internet.

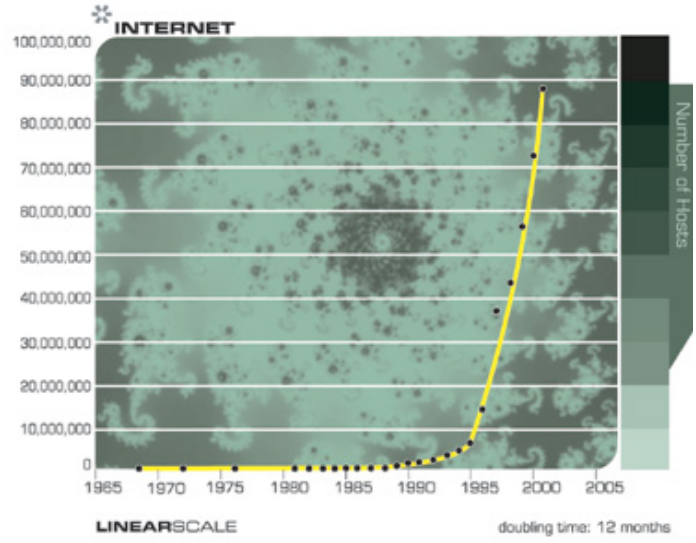


Table 2: The Number of Internet Hosts

Teams

In the 1890s, Frederick Taylor's work on time and motion studies introduced the assembly line and began a move toward large scale factories.

Henry Ford took it to the next step and started moving the parts on the assembly line so that people could stand in one location, wasting less time in unnecessary movement.

These advances were important, but they were mostly about people doing very specific, very repetitive tasks. Essentially, they were told what to do by their managers, watched by their managers, and they did what they were told.

Taiichi Ohno, the father of the Toyota Production System introduced a few new concepts to mass production:

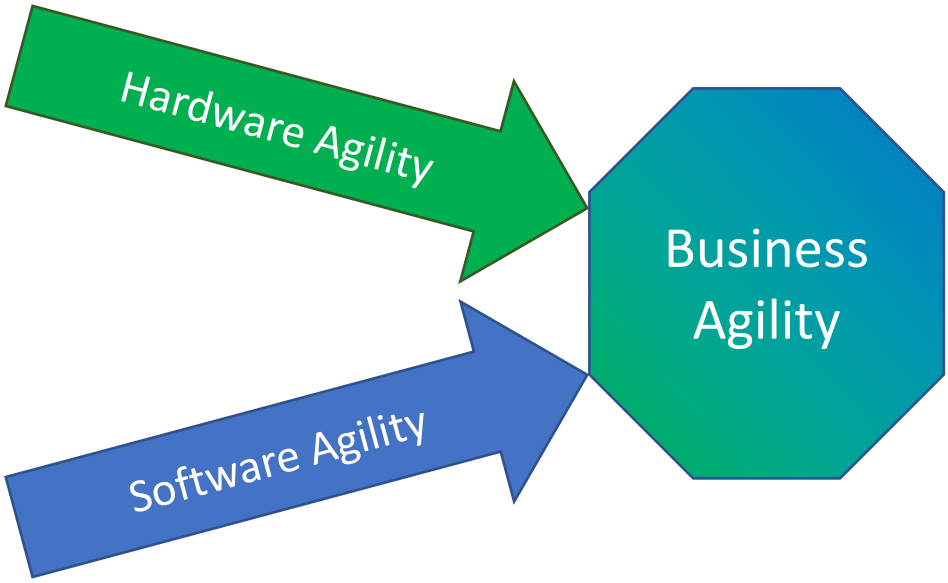
- Eliminate all unnecessary waste (including building inventory before you need it)
- Continuous improvement

To implement continuous improvement, Dr. Ohno proved that teams are far more effective at not only continuously improving, but he also proved that they were far more capable of taking direction from management and then making it happen, usually without direct supervision.

In 1985, Agile Manufacturing was created from the concepts of lean manufacturing and the advantages providing by ever-increasing technology. For example, it now became possible to custom-order automobiles on a regular basis (it still cost a little more but was no longer an extravagant expense).

In 2001, Agile Software Development was created, again from the concepts of lean manufacturing and advantages provided through more advanced programming languages (C, Java) and new tools to support product development (source code control, automated testing)

In 2007, the concepts of Agile Manufacturing and Agile Software Development merged into the term *Business Agility*, a term which encompassed using the principles of agility (empowered teams, working products, close collaboration with stakeholders and customers, and adapting to change rather than sticking to an out of date plan) in any environment. Agility is now used in IT environments, manufacturing environments, schools, government agencies including the military branches, and human resource departments).



The Agile Manifesto[1] and the Values of Agile Development

We are uncovering better ways of developing software by doing it and helping others to 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

The Principles Behind the Agile Manifesto[2]

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. Businesspeople 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.

[1] (Beck, et al., 2001)

[2] (Beck, et al., 2001)

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.

Scrum

Scrum was created by Jeff Sutherland at Easel Corporation, Princeton, in 1992. Over the next three years, with the help of his team at Easel as well as Ken Schwaber, Scrum was codified and released to the world in 2002 with in *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle. Scrum is a framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

The fundamental framework of Scrum is straight-forward. There are three roles (Product Owners, ScrumMasters, and Development Teams), three artifacts (Product Backlog, Sprint Backlog, and Product), and five events (Sprints, Sprint Planning, Daily Scrum, Sprint Review, and Sprint Retrospective).

Agile Values in the Scrum Framework

As a framework supporting Agile Development, the values and principles of Agile Development have strongly influenced the definition and the Scrum framework.

Agile Value	Scrum
Individuals and Interactions	Cross-functional teams with a transcendent purpose and the empowerment to make decisions about building the product.
Working Software	Sprints end with a shippable (production quality) product and Sprint Reviews focus on validating that progress.
Customer Collaboration	Customers and stakeholders are encouraged to work directly with Scrum teams rather than relying on documentation as a guide.
Responding to Change	The content and order of the Product Backlog (the list of work to be done on the product) can be modified at any time (with the exception that the content of an active Sprint is not affected).

Agile Principles in the Scrum Framework

Agile Principle	Scrum
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Short sprints and a Product Backlog ordered by value help to ensure high value first and potentially frequent demos for the customer.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	The content and order of the Product Backlog (the list of work to be done on the product) can be modified at any time (with the exception that the content of an active Sprint is not affected).
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.	The longest Sprint duration in Scrum is one calendar month, well within the shorter timescale preference recommended by the Agile principles.
Businesspeople and developers must work together daily throughout the project.	Scrum invites stakeholders to engage directly with the Scrum team through the Product Owner or the Development Team.
Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.	Scrum teams choose the best way to accomplish their work. They are formed as a cross-functional and self-organizing entity and expected to do what is necessary to accomplish their goals.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Scrum team members are, preferably, full time and co-located.
Working software is the primary measure of progress.	Sprints end with working product. Work is gauged as complete when deliverable to the customer.
Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.	Scrum teams determine the extent of their commitment every Sprint. In general, it is considered best practice for the team to base their forecast for a Sprint on the result of the previous Sprint.

Agile Principle	Scrum
Continuous attention to technical excellence and good design enhances agility.	During the Sprint Retrospective, the Scrum team is encouraged to identify ways to improve their work processes, adapting DONEness, and team practices.
Simplicity--the art of maximizing the amount of work not done--is essential.	Scrum is a very simple framework with three roles, five events, and three artifacts. Teams are encouraged to experiment and improve.
The best architectures, requirements, and designs emerge from self-organizing teams.	Scrum teams begin work with Product Backlog Items that are not fully detailed and progressively elaborate the items in time to do the actual work.
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Scrum incorporates a Sprint Retrospective at the end of every Sprint during which the team reflects on their performance thus far.

Quiz!

Please complete the following questions:
1. How does Scrum implement the Agile value of Customer Collaboration over Contract Negotiation (hint: you can find the Agile values and how Scrum implements them on page 6)?
2. How is the principle of "Simplicity" exemplified in Scrum (hint: you can find the Agile principles and how Scrum implements them on page 7)?

3. How does Scrum implement the value of "responding to change over following a plan?"

The Scrum Roles

In completing the work of the Sprint, the Scrum roles work together. The Product Owner helps the Development Team understand the work to be completed and works with the Development Team to resolve questions about product requirements. The Product Owner also optimizes the work that the Development Team completes, usually by ordering the work based on business value.

The Development Team is self-organizing, cross-functional, and accountable for the work they perform. The Development Team decides how they turn Product Backlog Items into releasable functionality. The cross-functionality of the Development Team is a key advantage for the team; it allows the Development Team to view all Product Backlog Items from multiple important perspectives; this provides for

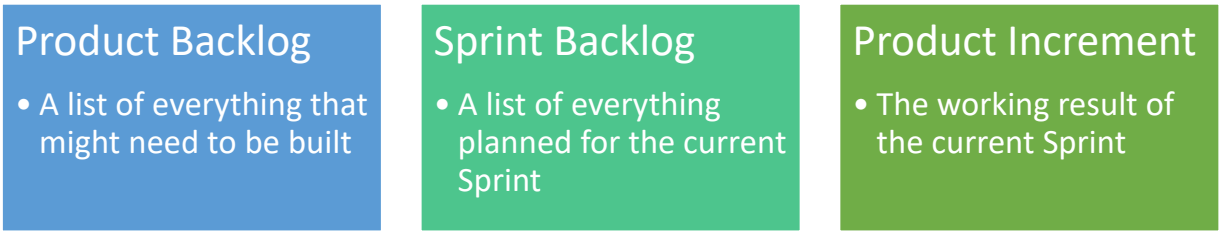
- A better understanding of the problem
- A better understanding of the solution, and
- More effective estimation and, thus, predictability.

The Development Team is also self-organizing, ensuring that the team decides how they want to do their work. This makes the team more accountable for the work that they perform – if they create the solution, they are most likely to make the solution work.

The Scrum Artifacts

The Product Backlog is an ordered list of everything that is known to be needed in the product. It is the **single source of requirements** for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.

A Product Backlog is **never complete**. The earliest development of it lays out the initially known and best-understood requirements. The Product Backlog evolves as the product and the environment in which it will be used evolves. **The Product Backlog is dynamic**; it constantly changes to identify what the product needs to be appropriate, competitive, and useful. If a product exists, its Product Backlog also exists.



As a product is used and gains value, and the marketplace provides feedback, the Product Backlog becomes a larger and more exhaustive list. Requirements never stop changing, so **the Product Backlog is a living artifact**. Changes in business requirements, market conditions, or technology may cause changes in the Product Backlog.

The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal. The Sprint Backlog is a **forecast by the Development Team** about what functionality will be in the next Increment and the work needed to deliver that functionality into a "Done" Increment.

The Sprint Backlog makes visible all the work that the Development Team identifies as necessary to meet the Sprint Goal. To ensure continuous improvement, it includes at least one high priority process improvement identified in the previous Retrospective meeting.

The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum. The Development Team modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint. This emergence occurs as the Development Team works through the plan and learns more about the work needed to achieve the Sprint Goal.

The Increment is **the sum of all the Product Backlog items completed during a Sprint** and the value of the increments of all previous Sprints. At the end of a Sprint, the new Increment must be "Done," which means it must be in useable condition and meet the Scrum Team's definition of "Done". An **increment is a body of inspectable, done work** that supports empiricism at the

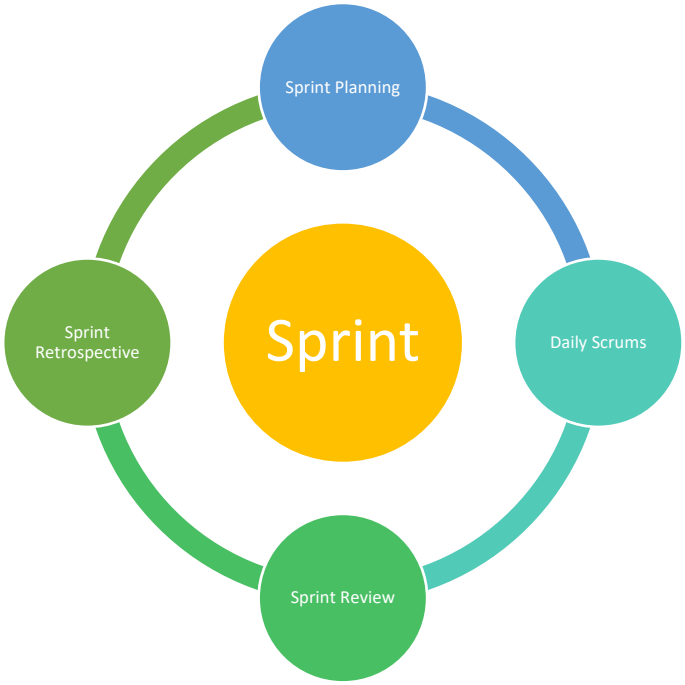
end of the Sprint. The increment is a step toward a vision or goal. The increment must be in useable condition regardless of whether the Product Owner decides to release it.

The Scrum Events

Sprints begin with Sprint Planning, during which the Product Owner and the Development Team discusses what each candidate Product Backlog Item means and whether or not it can be built in the current Sprint. The **Development Team makes the final decision** with regard to the total content of the Sprint. The end result of the Sprint Planning event is the Sprint Goal (discussed later) and the Sprint Backlog.

During the ensuing Sprint, the Development Team and the Scrum Master meet daily to discuss the condition of the Sprint. The Product Owner may join should they wish, but their involvement in the Daily Scrum is not mandated.

At the close of the Sprint, the Scrum Team performs a Sprint Review during which the result of the Sprint is reviewed and discussed, and next steps are planned. Following the Sprint Review, the team conducts the Sprint Retrospective to evaluate the team's performance during the Sprint and brainstorm ways to improve the team's capabilities.



Scrum is not a process, technique, or definitive method. Rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of your product management and work techniques so that you can continuously improve the product, the team, and the working environment.

Timeboxing

All Scrum events are timeboxed. This means that they have an absolute maximum length at which they are ended. Timeboxes are normally established before the event begins and help to ensure that

- An event stays focused on its goals
- An appropriate amount of time is spent in each activity
- Wasting of time is mitigated due to the hard stop at the end of the event.

Quiz!

Please complete the following questions:
1. Scrum is the most used agile framework in the world. What properties of Scrum most likely supports this outcome (i.e., why is it so popular)?
2. Which Scrum event runs as a fixed timebox (cannot be lengthened or shortened) as compared to a maximum timebox (could be shortened if the event goal is met)?
3. Why is it important that a Scrum team be self-organizing? What would happen if they were given specific direction by a manager or other outside authority?
4. ACME Corporation just launched five "Coding Teams" (teams that are coders only). What should be expected with regard to the quality of the resulting product?

The Scrum Values

There are five values that, when exemplified by the Scrum team, can result in a highly productive, extraordinarily capable team. The values are

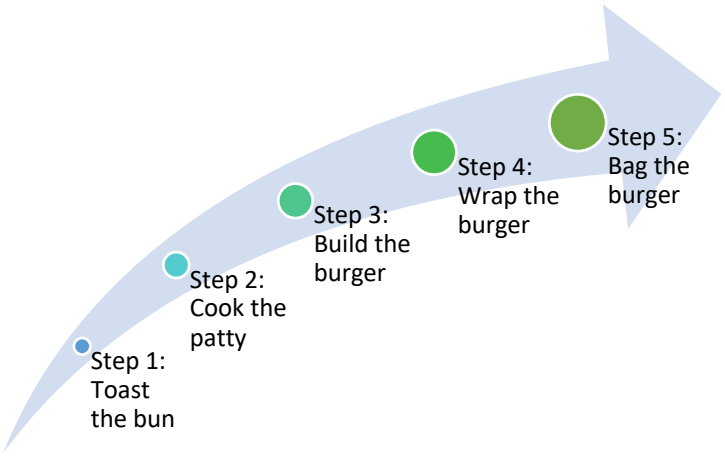
Focus is probably the most difficult of all the values to create on a team. The average office worker is interrupted once every 3 to 5 minutes and getting back to work after some interruptions can cost an average of 23.5 minutes![3] When creating an environment for teams, remember that once the Sprint begins, we avoid interrupting the team as much as possible.

Empirical Process Control

Imagine working at a fast food restaurant and making a hamburger. You actually only have to do one part of the overall work: you put the wrapped hamburger into a bag. Someone else has

[3] (Mark, Gudith, & Klocke, 2008)

already toasted the bun. Another person cooked the burger patty. Someone else put it all together and wrapped it.



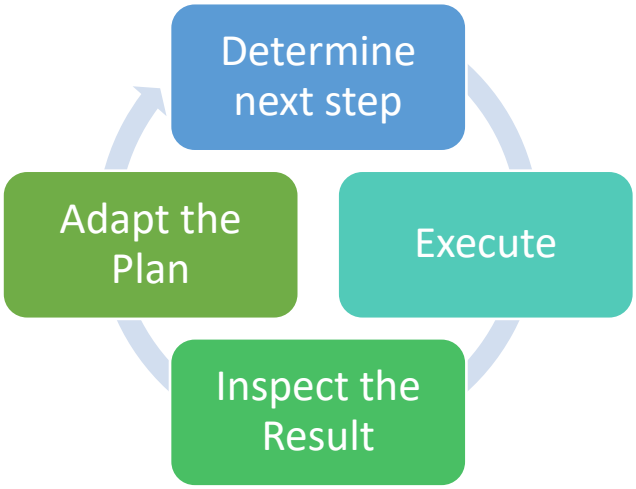
You are participating in an assembly line which is a great example of a DEFINED WORKFLOW. A defined workflow is exactly what it sounds like. The workflow is strictly designed. Each step follows the next in a perfectly linear fashion. This allows for greater efficiency and quality because the hamburger is the SAME product over and repeatedly. There's little to no variation (ok, one guy doesn't want pickles).

Imagine doing something like this with more complex work. Perhaps it is a custom product built for a single customer – for example, a customer software application or a policies and procedures manual for your company or even a sales team whose "product" is to generate sales for the company. Perhaps it's something built for an entire market like a smartphone app, an internet appliance, or even both.

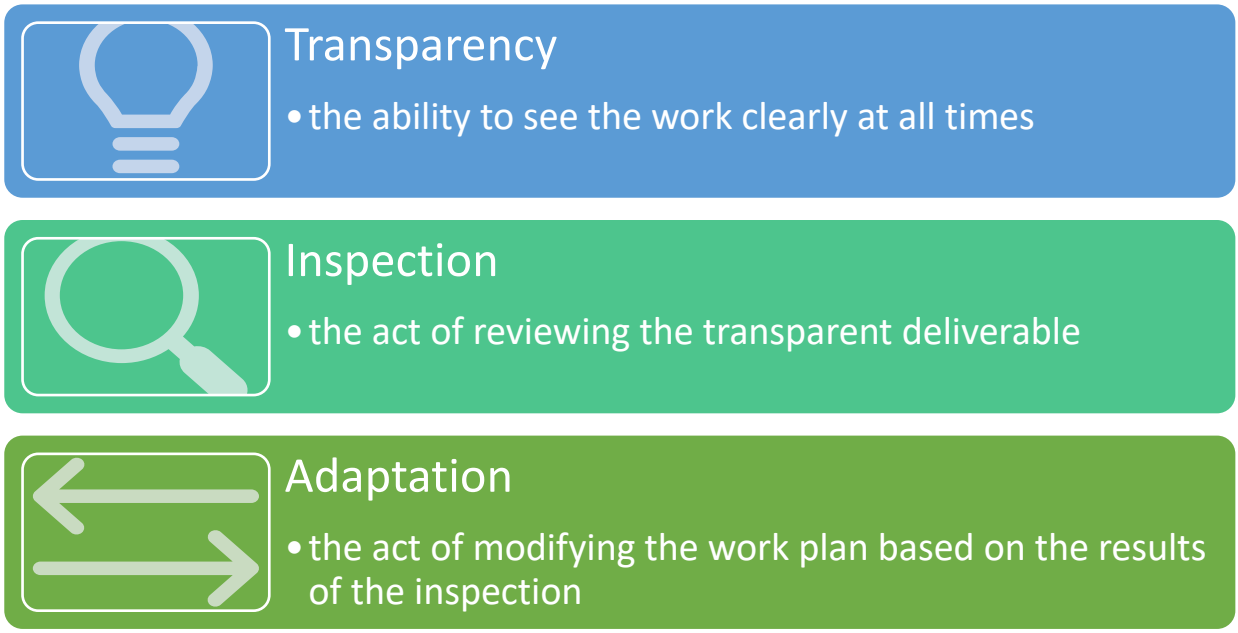
Using a defined workflow means that we would take the exact same steps every time we did whatever it is our team does. But what if every time we did our work, it was different? For example, EVERY hamburger we made was different than the last one. What if every time we did our work, something unexpected changed or didn't work the way it was supposed to? Every time we cooked the burger patty it had to be done differently. Could you use a defined workflow now? Remember, defined workflows work because NOTHING ever changes all that much.

Scrum uses an empirical workflow. This means that we do a little work and then we stop and reflect on what we accomplished. Some feedback is collected from stakeholders and customers and then we decide what we should work on next and get back to work. After a bit of time, we stop again and repeat the process.

The basic flow looks like this:



Scrum defines empiricism as requiring three actions (sometimes called “The Three Pillars”):



All of this comes together in Scrum as a series of inspect-and-adapt feedback loops driven by the Scrum events:

- Sprint Planning and Sprint Reviews inspect and adapt the PRODUCT
- Sprint Retrospectives inspect and adapt the TEAM
- Daily Scrum events inspect and adapt the SPRINT

Partial Implementation

It is, unfortunately, very common for organizations to implement a "Scrum-like" or "Scrum-hybrid." **This is usually an excuse for doing what the organization was doing before** but calling it Agile anyway. It also always underlines an existing dysfunction in the organization that the organization has no desire to correct or, perhaps, attempt to correct.

For example,

- An organization plans everything in advance and simply uses the Sprint as a vehicle for telling the team what they have to do. The excuse is often that the product is too complex to be built incrementally and must be completely planned up front. The reality is that complexity is the REASON for incremental development. The more complex something is, the more likely we are to build it wrong and waste time.

- A Scrum team decides to skip the daily Scrum because they just aren't "getting any value out of it." The reason for this is nearly always because the team members aren't really working like a team. They have implemented Scrum but are continuing to be a waterfall group – task-driven and everyone is working independently of everyone else.

Don't be fooled by Scrum-hybrid implementations. When we partially implement Scrum, we are likely not implementing the values of Agility – in this case, we must honestly ask ourselves if the change from waterfall to non-agility is really worth it.

Keep in mind, **all of the events in Scrum are part of a feedback loop** (e.g., Sprint Planning and Sprint Review loop on the product). Eliminate any one of the events and you disable a loop meant to incrementally improve the product or the team.

Likewise, **the values of Scrum are VERY important and easily ignored**. For example, when we decide that the team members with the testing skills are the ONLY team members allowed to test and, if the team doesn't finish everything by the end of the Sprint, it was the TESTER'S fault, we are ignoring the values of commitment, respect, and trust. Under these conditions, you needn't spend any effort attempting to improve your team's productivity. You should consider yourself lucky if they simply maintain their current levels of productivity.

Quiz!

Please complete the following questions:

1. **How does a defined workflow differ from an empirical workflow?**

Please complete the following questions:
2. What are the Scrum values?
3. What are Scrum's 3 Pillars of Empiricism?
4. How does Scrum exemplify the pillar of "transparency" regarding the product?
5. Your team just decided NOT do to Sprint Retrospectives anymore. What is a likely cause for this decision and what is the most likely result of not holding regular Sprint Retrospectives?

Scrum Roles

The Scrum Team

The Scrum Team consists of

- a Product Owner,
- the Development Team,
- and a Scrum Master.

Scrum Teams are

- self-organizing and
- cross-functional.

Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team.

Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity. The Scrum Team has proven itself to be increasingly effective for any complex work.

Scrum Teams deliver products iteratively and incrementally, maximizing opportunities for feedback. Incremental deliveries of "Done" product ensure a potentially useful version of working product is always available.

The Product Owner

The Product Owner is responsible for maximizing the value of the product or service resulting from work of the Development Team. In this context, “maximizing the value of the product or service” is about ensuring that the Development Team can

1. Build the greatest business value possible in the time budgeted, and
2. Avoid wasting time in activities that do not add to the value of the product or service.

How this is done may vary widely across organizations, Scrum Teams, and individuals.

The Product Owner is the sole person responsible for managing the Product Backlog. This includes:

- Clearly expressing (writing as well as explaining) Product Backlog items;
- Ordering the items in the Product Backlog to best achieve goals and missions;
- Optimizing the value of the work the Development Team performs;
- Ensuring the Product Backlog is visible and clear to all, and shows what the Scrum Team will work on next; and,
- Ensuring the Development Team understands items in the Product Backlog to the level needed.

The Product Owner may do the above work or have the Development Team do it. However, the Product Owner remains accountable.

The Product Owner is one person, not a committee. The Product Owner may represent the desires of a committee in the Product Backlog, but those wanting to change a Product Backlog item’s priority must address the Product Owner.

Rights of the Product Owner

The Product Owner has the following sole rights:

- The right to order the Product Backlog has he or she sees appropriate,
- The right to determine the value of a Product Backlog item,
- The right to determine the content of the Product Backlog,

- The right to determine when the product increment is ready to ship to the customer,
- The right to have their decisions about the product respected by the organization,

The Scrum Development Team

The Development Team consists of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint. A "Done" increment is required at the Sprint Review. Only members of the Development Team create the Increment.

Development Teams are structured and empowered by the organization to organize and manage their own work. The resulting synergy optimizes the Development Team's overall efficiency and effectiveness.

Development Teams have the following characteristics:

- They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality;
- Development Teams are cross-functional, with all the skills as a team necessary to create a product Increment;
- Scrum recognizes no titles for Development Team members, regardless of the work being performed by the person;
- Scrum recognizes no sub-teams in the Development Team, regardless of domains that need to be addressed like testing, architecture, operations, or business analysis; and,
- Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole.

Development Team Size

Optimal Development Team size between three and nine team members. This is small enough to remain nimble and large enough to complete significant work within a Sprint. Fewer than three Development Team members decrease interaction and may encounter skill constraints during the Sprint, causing the Development Team to be unable to deliver a potentially releasable increment. Having more than nine members requires too much coordination. Large Development Teams generate too much complexity for an empirical process to be useful. The Product Owner and Scrum Master roles are not included in this count unless they are also building the work of the Sprint Backlog.

The Rights of the Development Team

The Development Team has the following sole rights:

- The right to determine the size of a Product Backlog Item,
- The right to determine the solution for a Product Backlog Item,
- The right to set the content of the Sprint,
- The right to determine how the work in the Sprint gets done, and
- The right to self-manage the content of the Sprint.

Quiz!

Please complete the following questions:
1. Which role is responsible for the ordering of the Product Backlog?
2. Which role is responsible for the determining the value of a Product Backlog Item?
3. Which role is responsible for determining the content of a Sprint?
4. Which role decides when a product is ready for the customer?

Please complete the following questions:
5. Which role is considered the servant leader of the Scrum team?

The Scrum Master

The Scrum Master is responsible for promoting and supporting Scrum as defined in the Scrum Guide. Scrum Masters do this by helping everyone understand Scrum theory, practices, rules, and values.

The Scrum Master is a servant leader for the Scrum Team. The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.

The Rights of the Scrum Master

The Scrum Master has the following rights:

- The right to protect the Scrum team from interruptions, both internal and external,
- The right to be respected as an expert in Scrum,
- The right to have their opinion heard by management when suggesting changes to the organizational structure or process,
- The right to participate in Sprint Retrospectives and have their voice heard from the perspective of the effectiveness of the Scrum team, and
- The right to use the authority granted to them by the Scrum team to the benefit of the team and the organization.

Facilitation

Imagine sitting in a team meeting and, throughout the meeting, the conversation wanders without the team ever coming to an understanding of the problem they want to solve much less a solution for the problem. It's very likely you've experienced this.

> In a recent survey of 182 senior managers in a range of industries: 65% said meetings keep them from completing their own work. 71% said meetings are unproductive and inefficient. 64% said meetings come at the expense of deep thinking. 62% said meetings miss opportunities to bring the team closer together.[4]

Facilitation is about helping teams understand their common objectives and devising a common solution.

Scrum Masters can provide simple but powerful facilitation by doing things like:

- *Design and plan how the Scrum team will complete a Scrum event*. Different kinds of discussions call for different approaches. Decision making can be done in a variety of different manners. For example, a Scrum Master could design and plan the approach to Sprint Planning by recommending that the first part of Sprint Planning (when the Product Owner and the Development Team discuss Product Backlog Items and make decisions about the content of the Sprint) be done through ten-minute Q&A sessions about each backlog item followed by an up or down vote to include in the Sprint. Any down votes result in another five-minute follow-up and another vote. Three failed votes and the team agrees that the backlog item isn't ready for planning and moves on.

[4] (Perlow, Hadley, & Eun, 2017)

Scrum Event Worksheet

Scrum Event (circle one): Planning Daily Scrum Review Retrospective

Scrum Team: *Perfect Pug*

Sprint (name/date): *Milkbone Sprint (November 2019A)*

Topic	Goal	Approach	Timebox
Sprint Goal	*Go around the team and see if anyone wants to contribute to a Sprint Goal. Vote to determine goal.*	*Discuss and vote*	*10 minutes*
Retrospective Finding	*Discuss the retrospective finding(s) from the previous Sprint and decide how to incorporate into the Sprint.*	*Discuss, delegate into Sprint if more information is needed.*	*10 minutes*
Scope the Sprint	*Determine the PBI content of the Sprint*	*Q&A and up/down votes. Down requires more discussion. 3 downs excludes PBI*	*70 minutes (plus 15 minute break)*
Solve the content	*Determine how to solve each PBI*	*Breakouts (15 minutes) with report out until solution.*	*120 minutes (plus 15 minute break)*
Evaluation (how did it go?)			

Table 3: The Scrum Event Worksheet - you can find a printable copy of this worksheet in the Appendices section at the end of this handbook.

- *Manage Scrum events to the plan, modifying as needed.* Not every plan works. A good facilitator updates and improves their planning, learning from experience and what the team prefers. This includes making sure that everyone is properly heard, that every idea gets appropriate consideration, and MOST IMPORTANTLY that the team OWN their decisions once those decisions have been made.

- *Ensure that actions, once agreed upon, are carried out.* Not by carrying them out himself, but by ensuring that the Development Team members and even the Product Owner follow through on what they said they were going to do.

- *Support the decision-making process intentionally.* Teams that are properly formed have the skills they need to do the job, but they often lack the skills they need to make good, committed decisions. A good Scrum Master can help with decision-making:

    - Modified Borda – Everybody ranks their choices from least-favorite (which gets 1 point) to most favorite (which gets a point value equal to the number of options). For example, when there are seven options to choose from, each team members ranks the options from 7 (most favorite) to 1 (least favorite). Add up the total scores for each option. The team doesn't HAVE to go with the option with the most votes, but it certainly clarifies how everyone feels.

    - Multi-voting (using dot voting) – everybody gets several votes equal to half of the available options (for example, with ten options to choose from, everyone gets five votes). In round one, everyone votes and the options that finished in the top 40-50% move on to round two. Reduce everyone's vote count to half of the remaining options and do it again. By round three, you should be able to eliminate all but the winner. **Note:** sometimes the "winner" is obvious in round one. Don't keep doing voting rounds if the team is satisfied. The point of this exercise is to ensure that everyone feels that their option was heard, not necessarily agreed with.

    - Consensus decision – everyone gets an up/down vote. You can either go with the majority or the team could decide that a decision is so important that it can't move forward without a "super-majority" (60-75%) agreement.

Coaching

Coaching is about helping to develop the skills and abilities of your Scrum team in order to help them improve their performance. The goal is always to help the coachee (the person being coached) to discover answers to problems for themselves. Coaching generally involves techniques like

- Active Listening – be attentive, in-the-moment, and focused on what's being said and why it's being said. Allow for silences as the coaches gathers their thoughts and avoid "filling in the blanks" or trying to provide answers for the coachee.

- Powerful Questions – ask question that make the coachee think. Consider questions like "what else could you do in that situation?" or "if you could do anything you wanted, what would it be?" or "what's stopping you from moving forward?" This will help the coachee form their own opinions, find their own solutions, and take responsibility for their next actions.

Servant Leadership

As defined by its creator, Robert K. Greenleaf, servant leadership is about being a "servant first." You focus on the needs of others, especially your team, before you consider your own. You listen to other's and appreciate their perspective. You provide the support they need to be successful. You involve them in decisions and build a sense of community within your team.

An important point to remember – servant leadership is about focusing on people's needs, not their feelings. Don't avoid giving team members negative feedback when appropriate.

A servant leader helps the team understand and remember their vision and their goals. In Scrum, that's the role played by the ScrumMaster. The specific goals of the ScrumMaster, as a servant leader, are to help the team:

- continuously improve
- remain focused and avoid unnecessary interruption
- make sure that the Scrum framework is used effectively, this means:
    - keeping an INTENSE focus on the Sprint goal,
    - ensuring RADICAL collaboration,
    - inciting a HUNGER to crush the work in front of them, and
    - creating a universal EXCITEMENT when anyone on the team succeeds.

Attributes of a Servant Leader

What specific things can you to that will make you a good servant leader? Here's a list of ten:

1. *Listening* – You'll be a much more effective leader if you spend most of your time listening instead of talking. Give people your full attention and watch their body language when they speak. Avoid interrupting and provide feedback on what they say.
2. *Empathy* – Try to understand not only what people are saying, but what their perspectives and intentions are. Very few people speak so clearly and completely that everything they want to communicate comes through clearly and verbally. Consider WHY they might be saying what they are saying.
3. *Healing* – As the Agile principle says, "Give [your team] the environment and support they need and trust them to get the job done." Make sure your team has a healthful workspace[5] and is happy doing what they are doing.

[5] This implies that the workspace is clean and safe, that team member growth and development is being addressed, that individuals are recognized for their achievements, that team members are involved in decisions about the workspace, and that team members can deal with personal matters (like a phone call from their doctor) privately.

4. *Self-Awareness* – look at yourself, examine your own emotions and behavior, and consider how they affect your team. Make sure your behaviors align with your values. Ask for feedback on how well you are doing your job.
5. *Persuasion* – use persuasion to encourage people to act. ScrumMasters have no authority, so this is your primary tool for getting things done.
6. *Conceptualization* – keep looking to the big picture; remind your team of their vision and goals on a regular basis. Keep everyone focused, not just on the job, but on the destination.
7. *Foresight* – use what you know and what you've experienced to predict what's likely to happen soon. By understanding where your team is going, you can do a better job of helping them avoid mistakes. There are tools you can use to analyze the situation but remember to trust your experience and intuition.
8. *Stewardship* – lead by example, make your actions match your values, and take responsibility for the actions and performance of your team (which might mean getting yelled at by a stakeholder or customer on behalf of your team). Always remember the Golden Rule: "Do to others what you would want them to do to you."[6]
9. *Commitment* – be committed to your team and the people on it. Make sure everyone has what they need not only to get the job done, but to advance their careers and their skills. Convince management to provide the training your team needs, or the time they need to learn, or the tools they need to get the job done more effectively.
10. *Build Community* – give your team members a team to be proud of. This can be accomplished by allowing everyone to feel like they are a part of something greater than a team. Build a community. Organize social events (lunches, BBQs) and opportunities to talk about things other than work.

The Impediment Remover

The Scrum Master is also expected to remove impediments on behalf on the Scrum Team. For clarity's sake, let's look at some examples[7] of these impediments:

- Illness of team members
- Unforeseen and undesired changes in team composition
- Issues with the tooling of the Development Team

[6] Luke 6:31, King James Version

[7] (Overeem, 2016)

- Scarcity of skills
- Lots of technical debt
- Problems with suppliers
- Unavailability of the Product Owner
- Undesired pressure from management
- Conflict between team members
- Lots of unimportant meetings the Development Team has to attend
- Restrictions to the team environment
- An indecisive Product Owner

When dealing with the impediment, the Scrum Master should first decide if the impediment is really something that the Development Team needs to resolve on their own (e.g., “lots of technical debt”) as well as whether or not the impediment is actually in the way or has gone beyond the Scrum team’s ability to self-organize around it.

Sometimes dealing with impediments is about going to someone who can help (e.g., management). Sometimes you have to deal with the team members themselves (e.g., when there’s a conflict between two team members).

Methods a ScrumMaster can use to address impediments include:

- **Use a Sprint Goal** – A clear Sprint Goal is a useful instrument to determine if something is truly an impediment. If something prevents the team from achieving the Sprint Goal, then it is an impediment.
- **Understand the difference between 'blocks' and 'impediments' –** A block affects only a single task, whereas an impediment acts like a parachute, slowing down overall progress. Quite often the Development Team can fix 'blocks' themselves whereas impediments need to be fixed by the Scrum Master.
- **Improve transparency by using an 'Impediment Board' –** This can be a simple flip-over where the impediments are visualized. You can also add the impediments to the existing Scrum task board.
- **Keep track of fixed impediments –** This will provide great input for the Sprint Review and Sprint Retrospective.

- **Understand the organization's culture** – Understand how things get done in the organization. By choosing the right approach, difficult impediments can be tackled more easily.

- **Be brave and creative in removing impediments** – Be prepared to ask for forgiveness afterwards when you need to take bold decisions to ensure the Development Team's productivity.

- **Collaborate with the Product Owner** – Impediments will often be related to product management and collaboration with stakeholders and suppliers. The Product Owner is a key player in this area.

> **Can one person play multiple roles? Yes, and you may need to do so from time to time. However, be aware that switching between roles creates significant context switching, which means that any person switching between multiple roles is likely only 1/3 as productive as they might be playing a single role. So, do it only when you need to and trust that people playing a single role, when properly trained and coached, will always be more effective.**

How the Roles Interact

When building the work of the Product Backlog, the roles on the Scrum teams interact in a variety of ways:

The Product Owner provides interacts with and supports the Development Team by

- Ensuring that goals, scope, and product domain are understood by everyone on the Scrum Team as well as possible by being the voice of the customer,

- Answering questions about priority and requirement,

- Prioritizing work,

- Inspecting work and making suggestions, and

- Determining when work is DONE and ready to deliver.

The Development Team interacts with and supports the Product Owner by

- Indicating how much work can be done during the Sprint,

- Identifying solutions for building the work,
- Determining WHICH solution and HOW the work will be done in the Sprint,
- Advising when a tradeoff can be made that improves ROI, and
- Providing working, potentially shippable product by the end of the Sprint.

The Scrum Master interacts with and supports the Product Owner by

- Ensuring that goals, scope, and product domain are understood by everyone on the Scrum Team as well as possible;
- Finding techniques for effective Product Backlog management;
- Helping the Scrum Team understand the need for clear and concise Product Backlog items;
- Understanding product planning in an empirical environment;
- Ensuring the Product Owner knows how to arrange the Product Backlog to maximize value;
- Understanding and practicing agility; and,
- Facilitating Scrum events as requested or needed.

The Scrum Master interacts with and supports the Development Team by

- Coaching the Development Team in self-organization and cross-functionality;
- Helping the Development Team to create high-value products;
- Removing impediments to the Development Team's progress; and,
- Facilitating Scrum events as requested or needed.

The Scrum Master serves the organization in several ways, including:

- Leading and coaching the organization in its Scrum adoption;
- Planning Scrum implementations within the organization;
- Helping employees and stakeholders understand and enact Scrum and empirical product development;
- Causing change that increases the productivity of the Scrum Team; and,

- Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organization.

Quiz!

Please complete the following questions:
1. Which role is responsible for supporting the Scrum team through coaching and facilitation?
2. The Scrum Master keeps talking about the vision for the product and why its so important for the team. Which attribute of a servant leader is she exemplifying (hint: you can find the attributes of a servant leader on page 26)?
3. Which role determines the solutions to be built during the Sprint?
4. What's the largest allowable size of the Scrum Development Team?
5. The Scrum Master listens intently to a developer and doesn't provide a single answer to the problems they are discussing! The Scrum Master just keeps asking more questions. What technique is the Scrum Master using?

Please complete the following questions:
6. **Bonus question: with regard to the previous question, what is the Scrum Master trying to accomplish behaving the way they are?**

How Scrum Changes Organizations

Scrum forces organizations to change the way they work. Why? Because

- Scrum emphasizes FOCUS as a value. This creates attention to the number of projects being worked on at the same time, how individuals interact when they might be causing an interruption, how non-critical defects are managed during a Sprint, where day-to-day decisions are made in the organization. Scrum Masters head off the "Hey, do you have a second…?" at-any-time questions in favor of allowing questions only during pre-planned "downtimes" or breaks.

- Agile emphasizes incremental development. This causes organizations to plan more frequent inspections and positions the stakeholders much more closely to the development teams. Stakeholders and customers, formerly used to "dumping requirements" and getting regular updates, are now being invited to live demos and feedback sessions on a regular basis.

- Agile emphasizes changing plans when changes are appropriate. This causes organizations to create long-term plans, but devise mechanisms for quick and frequent re-planning. For example, Cisco Systems has, for several years, included an "Agile Commit," a quarterly summit of managers and Product Owners to reassess the business justifications of ongoing projects and 1) continue them, 2) redirect them, or 3) cancel them. Redirection and cancellations are immediate. Teams might literally be reallocated to new projects by the conclusion of the event.

Where Did the Project Manager Go?

In Scrum, the traditional responsibilities of the Project Manager have been split between the Product Owner, the Scrum Master, and the Development Team.

The Product Owner makes decisions about scope, priority, and budget. When provided a budget, it is the Product Owner that decides what gets built first and how to position tradeoff decisions when scope needs to change within the same budget. The Product Owner is also responsible for providing product updates to management on a regular basis (usually during Sprint Reviews).

The Scrum Master provides the day-to-day monitoring of the work that the Development Team is performing, helping to ensure that processes are followed, tasks are updated by the team members, the Development Team "keeps their eyes on the prize" (the Sprint Goal), and management is kept informed of the status and condition of the Scrum team.

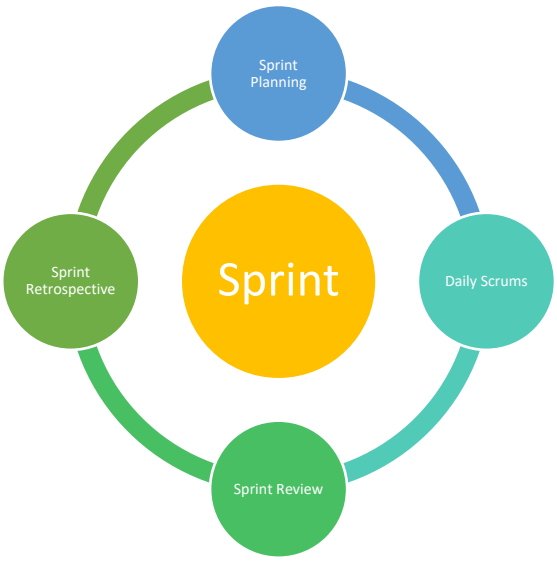
The Development Team provides the task breakdown, estimation of both tasks and Product Backlogs, and determines how much Product Backlog content can be done in a single Sprint (though they can modify that forecast during the Sprint should it prove appropriate).

The absence of a Project Manager in Scrum does not imply that the Project Manager role has no value. Projects with several teams in multiple locations where the overhead of keeping all of the teams communicating and aligned frequently benefit from a Project Manager.

Scrum Events

Scrum has five events defined in the framework:

- The Sprint
- Sprint Planning
- The Daily Scrum
- Sprint Review
- Sprint Retrospective



All Scrum events are timeboxed. This means that they have a maximum suggested length and that any Scrum event, once planned NEVER runs longer than the timebox. Timeboxing is essential because timeboxed events:

- Focus the team on a SPECIFIC objective for a SPECIFIC period,
- Focuses the team on a goal rather than a task,
- Creates a “leaner” approach to doing work,
- Provides for visible progress,
- Provides for accountability.

The Scrum Events				
Scrum Event	When?	Who?	Outcome?	Timebox?
The Sprint				
Sprint Planning, Part I				
Sprint Planning, Part II				
Daily Scrums				
Sprint Review				
Sprint Retrospective				

The Scrum events that occur at the beginning and the end of the Sprint require the entire Scrum team (Product Owner, Development Team, and Scrum Master). In the United States and many other countries, the days least likely to have everyone on the team working and available are Mondays and Fridays. Therefore, consider starting your Sprints on Wednesdays or Thursdays and ending them on Tuesdays or Wednesdays.

Scrum Roles During the Scrum Events

During the Sprint:

- Product Owner – checks in with the Development Team as frequently as the team needs, ensuring understanding and validating what's been built so far. The Product Owner will spend much of the Sprint time
    - Liaising with stakeholders, customers, and users
    - Preparing for Product Backlog Refinement
    - Tuning the Product Backlog to maximize ROI
- Development Team – builds the product and ensures DONEness; asks questions of the Product Owners, stakeholders, customers, and users when necessary; stays focused.
- Scrum Master – supports the team by removing impediments to progress; coaching and guiding the team; protecting the team from unnecessary interruption.

During Sprint Planning, Part I:

- Product Owner – help determine the Sprint Goal; identify and explain PBIs that will help the team achieve the Sprint Goal and maximize the value produced by the Development Team.
- Development Team – help determine the Sprint Goal; identify and understand PBIs that will help the team achieve the Sprint Goal; decide which PBIs are forecast for completion in the current Sprint.
- Scrum Master – facilitate as required or requested.

During Sprint Planning, Part II:

- Product Owner – helps the Development Team by further clarifying Product Backlog Items and making tradeoff decisions when the Development Team cannot do everything they decided to do in Sprint Planning, Part I.

- Development Team – determines an overall design for the product to incorporate the PBIs forecast in Sprint Planning, Part I; solving and identifying the work necessary to turn the forecasted PBIs into product; ensures that the work that will be done first is sliced into small enough tasks and ready to be immediately started; should be able to explain their approach to the entire Sprint by the end of Sprint Planning.

- Scrum Master – facilitates as requested or required.

During the Daily Scrum:

- Product Owner – involvement not mandated by the Scrum Guide

- Development Team – update each other on progress and impediments; determine if the Sprint Goal is going to be achieved by the end of the Sprint.

- Scrum Master – facilitate as requested or required.

During the Sprint Review:

- Product Owner – make a final determination regarding DONEness of each PBI in the Sprint; update attendees on changes in timeframe, budget, priority, staffing; update attendees on the state of any release dates.

- Development Team – provide a demonstration of completed functionality during the Sprint; report on development challenges faced by the team during the course of the Sprint and how those challenges were resolved; identify product improvements to the Product Owner to be added to the Product Backlog; field questions from the Product Owner and any other stakeholders about the product functionality.

- Scrum Master – facilitate as requested or required.

During the Sprint Retrospective:

- Product Owner and Development Team – discuss how the team could be more effective in the next Sprint.

- Scrum Master – facilitate as requested or required; ensure that follow-up commitments are upheld.

Quiz!

Please complete the following questions:
1. What's the longest a Sprint Planning event can be?
2. Who must attend the Daily Scrum?
3. Who determines the actual content of the Sprint?
4. Who determines the Sprint Goal?
5. Who works to remove impediments during the Sprint?
6. Who owns the right to set the business value of a Product Backlog Item?

Please complete the following questions:
7. Bonus Question: What are my options as a Development Team member if the Product Owner requests a very small change to my work during the Sprint?

When planning a Sprint, don't worry about the days of the Sprint (in other words, you don't need to plan Day 1, Day 2, Day 3, and so on). Just plan the Sprint. Tasks will get done when the team is ready to do them.

Scrum Artifacts

There are three artifacts in the Scrum framework. They are:

- The Product Increment,
- The Product Backlog,
- The Sprint Backlog, and

The Scrum Artifacts		
Artifact	Owner	Purpose
The Product Increment		
The Product Backlog		
The Sprint Backlog		

Products Require a DONEness Definition

The Product Increment is defined as the outcome or deliverable of the Sprint. For the Product Increment to be valid, it must be potentially shippable (which also means, to the best of the ability of the Development Team – without defect). Scrum recommends the creation of a DONEness definition to ensure that the quality of the Product Increment is as good as the Development Team can make it through a clear and shared definition of what it means to be DONE.

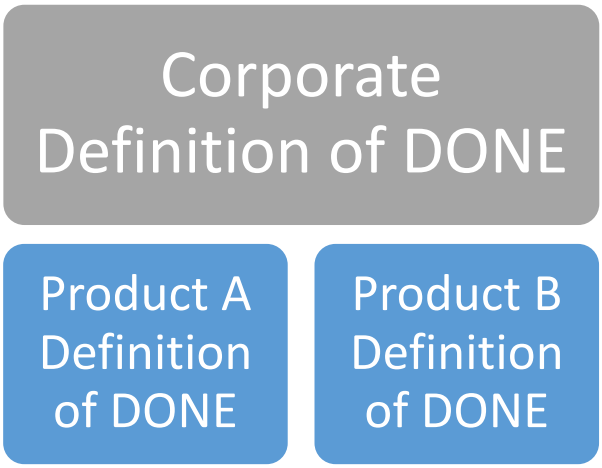
DONEness definitions apply on a Product Backlog Item basis and frequently includes items like:

- Works as the Product Owner wishes,
- Integrated with the rest of the existing product and nothing broke,

- There are no known errors,
- All documentation has been created,
- All approvals have been achieved, and
- All standards and regulatory required have been satisfied.

There may be additional items, as needed by the team.

It is the Scrum team that creates the DONEness Definition and it is, likewise, the Scrum team that owns what they create. When multiple Scrum teams work together on a single product, it is recommended that they use the same DONEness Definition. For example, an organization might have a DONEness definition that applies to all work in the corporation. This layer often includes document naming standards; ISO-9000 or CMM/I methods; or other corporate-wide standards. The same organization, however, could also include a product-level definition of DONE that all teams working on that product will employ. The product level DONEness definition then adds to DONEness items to the existing standard that are specific to the product itself. For example, Product A may be regulated by the Securities Exchange Commission (SEC) and have considerably more artifacts and deliverables required to meet the regulatory standards. Product B may be a tool for another department that has only modest additional DONEness criteria.



When teams working on the same product use the same DONEness definition, it improves

- Expectations of work results between teams,
- Improves overall product quality,
- Ensures the potential ship-ability of the combined product after each Sprint.

Technical Debt

When work is done on a product to a degree of quality that is less than appropriate for the product, technical debt is created. For example, if in building student management system for a university, the financial system functionality is not effectively tested, technical debt accumulates. Then, when the student management system goes live, it would not be

unexpected to discover that none of the student was billed the proper amount for their classes. Mistakes like this can, and have, led to the bankruptcy of the unlucky business.

"Technical debt is the difference between the quality that must be delivered and the quality that is actually delivered."

- Jim Schiel

Technical debt is, a manner speaking, the difference between the quality that must be delivered and the quality that is actually delivered. It's called "debt" because, one way or the other, the Development Team is going to have to pay it back later. It's also universally true that paying it back later will cost a lot more than if the problem had been found and fixed at the time it was created. For example, testing for and fixing the financial integration problems in the student management system would have most certainly been less expensive than the tuition shortfall because of the problems.

Because of the cost and potential dangers, it is recommended that technical debt be avoided whenever possible. This means:

- Comprehensive DONEness definitions that are improved whenever a significant error escapes the Scrum team's efforts,
- A PBI should never be considered DONE if there is a known defect of any kind.

It is also recommended that certain development practices be used by Scrum teams to help detect and help avoid debt in the first place:

- Low Work in Process – Scrum teams should never work on more than 1 or 2 PBIs at the same time and should collaborate (sometimes called "swarming") on items to get them done quickly. Fewer concurrent PBIs means less complexity and fewer mistakes.
- Small PBIs – Scrum teams should try to never work on PBIs that will take more than three days of effort. By keeping PBIs small, complexity and rework (debt) is reduced.
- Continuous Integration – many defects (debt) are found during the integration of new and changed features into the rest of the product. Frequent integrations mean that these defects are found sooner and are very easy to fix.
- Pair Programming – two team members work together on the same artifact (code, tests, documentation, specifications); pairing improves the quality of the artifact immediately with very little additional effort.
- Automated and Continuous Testing – as with continuous integration, defects can be found very rapidly (and when they are at their least expensive to fix) if testing is done continuously.

Product Backlog Items (PBIs)

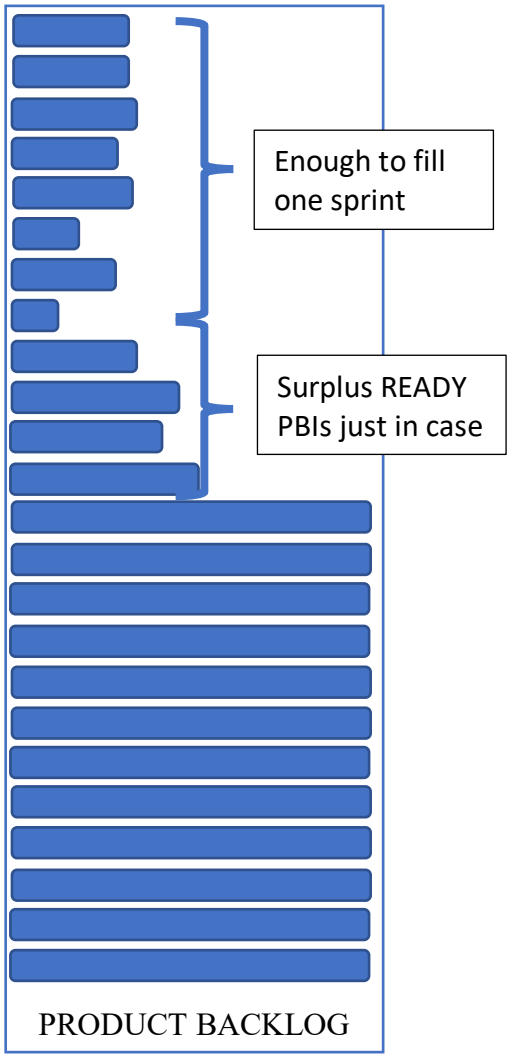
Product Backlog Items describe the features, functions, requirements, enhancements, and fixes that the Product Owner wants added to the Product. As requirements rarely stop changing, the Product Backlog Items are constantly in flux (being added, changed, moved elsewhere in the Product Backlog, or removed entirely from the Product Backlog).

Product backlog items have the following attributes:

- Description – the short textual description of the feature, function, fix, requirement, etc.,
- Order – where in the Product Backlog is the item placed (first, second, tenth, one-hundredth, etc.),
- Estimate – what's the cost of the item (usually in time, effort, or size), and
- Value – what's the value of the item.

Product backlog items sometimes also include:

- Acceptance criteria – detailed requirements
- Test descriptions – which help validate that the product backlog item is completed.



Product Backlog Refinement

Refining the Product Backlog is something that most Scrum teams do to one degree or another. Good backlog refinement generally

- leads to a better understanding of the Product Backlog Items,
- leads to better solutions for the Product Backlog Items,
- reduces waste caused by excess complexity,
- keeps the team more focused on value,
- allows for more targeted value-based ordering, and

- results in more productive Sprints.

During Product Backlog Refinement, the following activities are most likely to happen:

1. Discussing large Product Backlog Items to better understand them.
2. Slicing large Product Backlog Items into smaller PBIs.
3. Sizing the smaller PBIs.
4. Assigning business value to the smaller PBIs.
5. Ordering the smaller PBIs in the context of the rest of the Product Backlog.

When completed properly, Product Backlog Refinement will result in a Product Backlog that has enough very small, ready PBIs "at the top" to ensure that the Development Team can plan one full sprint plus some additional PBIs just in case the Development Team can do more work or if some of the work they plan into the sprint cannot be completed and has to be jettisoned.

Product Backlog Refinement is scheduled by the Scrum team as needed (it is not a Scrum event but is considered a best practice). Often, the event is scheduled some time in the middle of the Sprint in order to make the Product Backlog ready for the next sprint.

Avoid trying to solve and plan Product Backlog Items during Backlog Refinement. Backlog Refinement is a time to understand Product Backlog Items so that we can break (slice) them into smaller and simpler pieces of work. At the time of refinement, we don't know for sure that every Product Backlog Item we discuss will be built. So, you're wasting your time solving and tasking them out during refinement.

Extra: User Stories

User Stories are a very popular format for writing Product Backlog Items. They are NOT part of Scrum and are, therefore, a good practice, but not a requirement for writing PBIs.

User Stories are powerful in terms of keeping the product backlog item focused on the user instead of on the functionality. For example, the user story

As a student, I want to add a class to my course schedule so that I can create the most efficient use of my time.

Is a very different PBI than

Add class to student schedule.

The user story tells you what needs to happen, for whom, and why it's important to them. The second, more basic PBI simply tells you to add a course to the student's course schedule. We lose why it needs to be done and what's important.

Of course, there's not a ton that a user story can add to

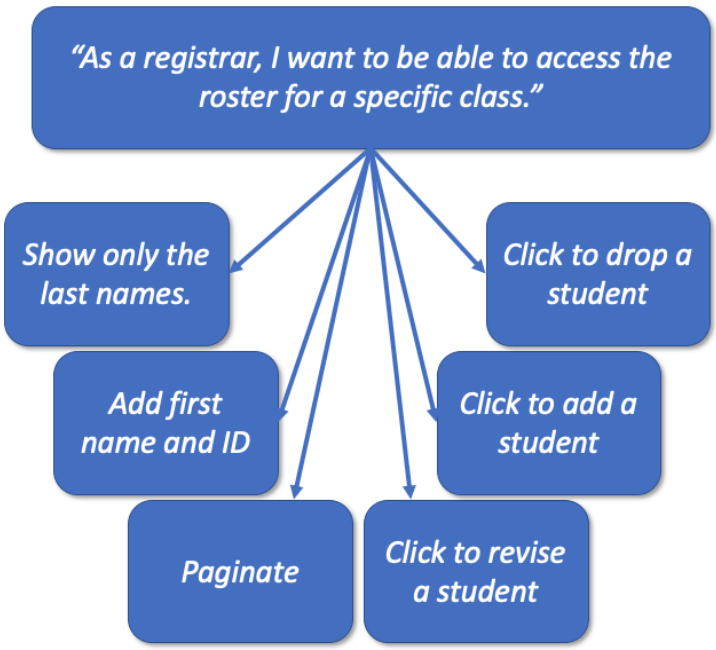
User logon

And that PBI might be more than enough as written.

So, not all PBIs need to be user stories but, in Scrum, ALL user stories are PBIs.

Extra: Slicing Product Backlog Items

When you reduce the size of a Product Backlog Item, you also reduce the complexity. This makes the smaller Product Backlog Item not only require less effort to build, it also makes the backlog item less risky.



To slice a Product Backlog Item into smaller pieces, one must first reach a shared understanding of the backlog item between the Product Owner and Development Team as well as, potentially, the users and the stakeholders. By engaging in a structured conversation, you will not only achieve a clearer understanding, but you will also easily find new ways to slice the backlog item into smaller items.

When slicing product backlog items, remember some simple rules:

- ✓ **You will rarely know everything the user wants before you start building. The user doesn't know either.**
- ✓ **A product backlog item should deliver production quality and value but doesn't have to be ready to go to production. In other words, you can build a very simplistic search capability that works (and, from a quality perspective, could go to production) in order to get more information from the user. Then, using that information, you build more.**
- ✓ **Never assume you're going to need to build some functionality the user hasn't asked for. If the user isn't asking for it, they might not need it.**
- ✓ **Never assume you're going to need to build some functionality for future use. The user will ask for it when they know they need it.**

This table shows some examples of ways to slice user stories.

Original User Story	Slices
As a university registrar, I want to be able to access the roster for a specific class. *(note: "access" usually means a lot more than "display;" clarify before continuing – once they have the roster, what do they want to be able to do next?)*	Let's start small… • I want to list the last names of the students in a class. ○ This will get the listing started and allow us to ensure that the user likes the basic UI approach. • Add student's first name and student ID to the listing. ○ Once we know the user likes the UI, we can supplement with more data to get the view they want to see. • Paginate the student listing. ○ During a demonstration of the functionality, the registrar asked for the ability to page forward, page backward, etc. She didn't want the entire student list as one long scrolling window.

	Now we have basics working, we can start adding more advanced functionality. • Once I have the student roster, I want to be able to revise a student. • From the student roster, I want to be able to add students • From the student roster, I want to be able to drop students.
As a bank account manager, I want to open a new account for a customer.	Start small… • I want to open a zero-balance savings account for an adult with no background check or address verification (not legal, but it lets us easily get something up and running for the customer to see). • Add balance requirement of $50 minimum. • Add address verification. • Do background check through automated system. • More?
As a radiology technician, I need to be able to search for patients currently in the hospital.	Start small… • Search for patients only by patient ID and display last and first name. We demonstrate this for the user and they like it, but also want to see patient location (where are they now) as well as the attending doctor's name • Add patient ID and attending doctor. • Add patient location (perhaps this took a little more work and was split out separately?) Another demonstration and the user says that they would like the list of patients sortable by both last name and patient id. • Add ability to sort by last name. • Add ability to sort by patient id.

Extra: Story Point Estimation

Story points, like user stories, are a great idea, but not part of Scrum and therefore something that Scrum teams can use or not use at their convenience. Story point estimation is a form of relative sizing. For example, compare the squares below. If I told you that the leftmost square had a "size" of 1, what size would you assign to the rightmost square?

If you said something between 3 and 5 (it's EXACTLY 4), you'd be in line with nearly every other sighted person who looked at them.

And how did you compare them? Of course, by area.

Comparing two items and making approximately, relative guesses about how they differ is something that the human brain is VERY adept at. If we apply that same comparison capability to the typical product backlog item, we often see the same "adept-ness" coming through in the estimations.

Obviously, we can't compare a PBI's "area," but we can look at

- Complexity – how much is there to do?
- Risk – how many things can go wrong?
- Bigness – how BIG is the overall effort (for example, rebranding an entire web site one page at a time – not complex, but quite big).

One other step we take to improve consistent estimation is that we specify the sizes that the team is able to use. In other words, how long do you think it would take for a team to size consistently if they could use any integer between 1 and 100? Imagine the arguments about a size of 20 and a size of 22!

We limit the sizes a team can use, sometimes using a modified Fibonacci sequence (1, 2, 3, 5, 8, 13, 20, 40, 100), sometimes using a doubling sequence (1, 2, 4, 8, 16, 32, 64, 128), and sometimes using a custom variation. In the practice of estimation, consistency is far more valuable than precision. If your team uses a consistent sequence of sizes, their estimates will work quite well.

Quiz!

Please complete the following questions:

1. **Which Scrum artifact details the content of the Sprint?**

Please complete the following questions:
2. Why is it so important to avoid/reduce technical debt?
3. We've written and put value on some new Product Backlog Items. As a Development Team, what's the remaining attribute that WE must provide?
4. Which role is responsible for identifying Product Backlog Item value?
5. What are the attributes of a Product Increment?

Sprinting

Sprints are a time-box of one month or less during which a "DONE", useable, and potentially releasable product Increment is created. Sprints have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.

Basic Structure

Sprints contain and consist of the Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective.

Sprint Length

Sprints are limited to one calendar month. When a Sprint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase. Sprints enable predictability by ensuring inspection and adaptation of progress toward a Sprint Goal at least every calendar month. Sprints also limit risk to one calendar month of cost.

Cancelling a Sprint

A Sprint can be cancelled before the Sprint time-box is over. Only the Product Owner has the authority to cancel the Sprint, although he or she may do so under influence from the stakeholders, the Development Team, or the Scrum Master.

A Sprint would be cancelled if the Sprint Goal becomes obsolete. This might occur if the company changes direction or if market or technology conditions change. When a Sprint is cancelled, any completed and "Done" Product Backlog items are reviewed. If part of the work is potentially releasable, the Product Owner typically accepts it. All incomplete Product Backlog Items are re-estimated and put back on the Product Backlog.

Sprint Planning

The work to be performed in the Sprint is planned at the Sprint Planning. This plan is created by the collaborative work of the entire Scrum Team.

Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches the Scrum Team to keep it within the time-box.

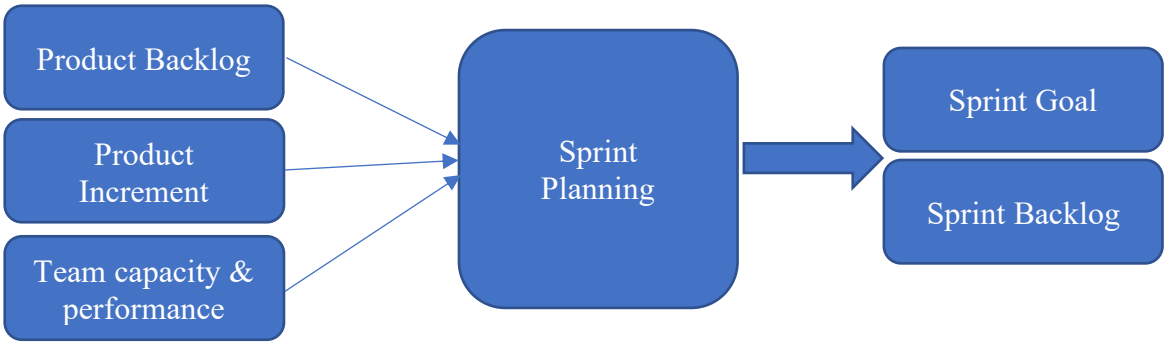
Sprint Planning answers the following:

- What can be done this Sprint?
- How will the work get done?

Part One: What can be done this Sprint?

The Development Team forecasts the functionality that will be developed during the Sprint. The Product Owner discusses the objective that the Sprint should achieve and identifies the Product Backlog Items needed to achieve the goal. The entire team collaborates on understanding the work of the Sprint.

The input to this meeting is the Product Backlog, the latest product Increment, projected capacity of the Development Team during the Sprint, and past performance of the Development Team. The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team. Only the Development Team can assess what it can accomplish over the upcoming Sprint.



During Sprint Planning the Scrum Team also crafts a Sprint Goal. The Sprint Goal is an objective that will be met within the Sprint through the implementation of the Product Backlog, and it provides guidance to the Development Team on why it is building the Increment.

Part Two: How will the work get done?

Having set the Sprint Goal and selected the Product Backlog items for the Sprint, the Development Team decides how it will build this functionality into a "Done" product Increment during the Sprint. The Product Backlog items selected for this Sprint plus the plan for delivering them is called the Sprint Backlog.

The Development Team usually starts by designing the system and the work needed to convert the Product Backlog into a working product Increment. Work may be of varying size, or estimated effort. However, enough work is planned during Sprint Planning for the Development Team to forecast what it believes it can do in the upcoming Sprint. Work planned for the first days of the Sprint by the Development Team is decomposed by the end of this meeting, often to units of one day or less. The Development Team self-organizes to undertake the work in the Sprint Backlog, both during Sprint Planning and as needed throughout the Sprint.

The Product Owner can help to clarify the selected Product Backlog items and make trade-offs. If the Development Team determines it has too much or too little work, it may renegotiate the selected Product Backlog items with the Product Owner. The Development Team may also invite other people to attend to provide technical or domain advice.

The Sprint Goal

Sprint Goals focus the Scrum Team on achieving something specific in the Sprint. Goals help the team make decisions and sets a clear bar for team achievement.

Sprint Goals can be things like:

- Complete five PBIs and fix all production defects
- Create added value for the customer in the very first sprint
- Create a deliverable that can be given to the customer in one sprint
- Kill the bugs!!! (this is a fun way to motivate the team to remove all remaining, known defects)
- Create more value by correcting our shortfall in specific coding skills

Sprint Goals can be anything the Scrum Team wants them to be but should generally be focused on something that directly or indirectly creates additional value (a better product, a better team, etc.).

During the Sprint, the Sprint Goal has the following effects:

- No changes are made that would endanger the Sprint Goal;

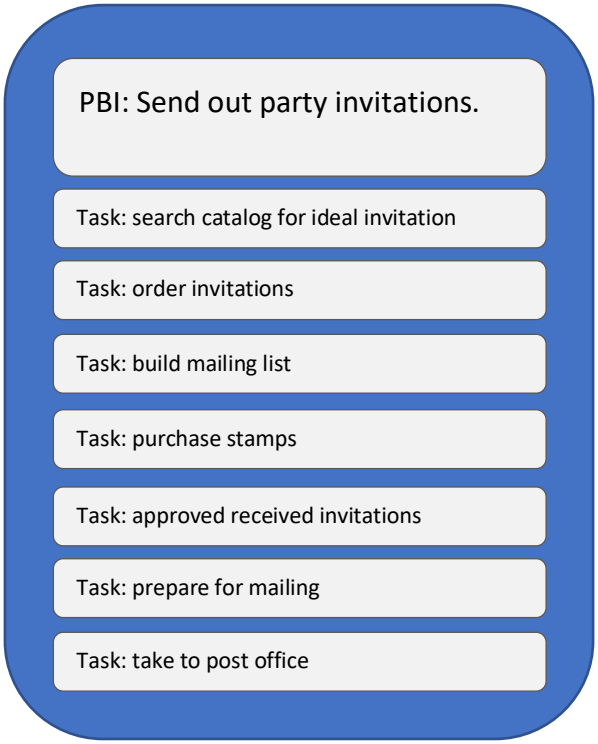
- Quality goals do not decrease; and,
- Scope may be clarified and re-negotiated between the Product Owner and Development Team as more is learned.

The Sprint Backlog

The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal. The Sprint Backlog is a forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a "Done" Increment.

The Sprint Backlog makes visible all the work that the Development Team identifies as necessary to meet the Sprint Goal.

The Sprint Backlog contains enough detail that changes in progress can be understood in the Daily Scrum. The Development Team modifies the Sprint Backlog throughout the Sprint.



As new work is identified, the Development Team adds it to the Sprint Backlog. As work is performed or completed, the estimated remaining work is updated. When elements of the plan are deemed unnecessary, they are removed. Only the Development Team can change its Sprint Backlog during a Sprint. **The Sprint Backlog is a highly visible, real-time picture of the work that the Development Team plans to accomplish during the Sprint, and it belongs solely to the Development Team**.

Daily Scrums

The Daily Scrum

- Is a daily, 15-minute time-boxed event during which the Development Team plans work for the next 24 hours,
- optimizes team collaboration and performance by inspecting the work since the last Daily Scrum and forecasting upcoming Sprint work, and
- Is held at the same time and place each day to reduce complexity.

The Development Team uses the Daily Scrum to inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog.

The structure of the meeting is set by the Development Team and can be conducted in different ways if it focuses on progress toward the Sprint Goal. Some Development Teams will use questions, some will be more discussion based. Here is an example of what might be used:

- What did I do yesterday that helped the Development Team meet the Sprint Goal?
- What will I do today to help the Development Team meet the Sprint Goal?
- Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

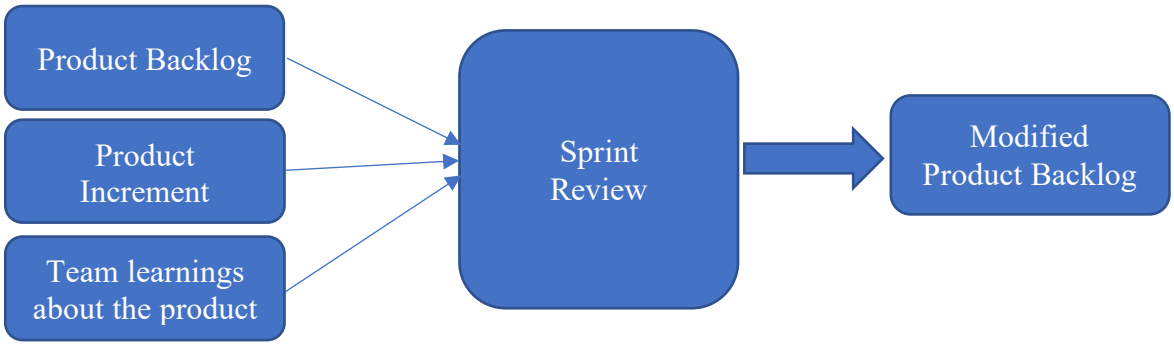
The Development Team or team members often meet immediately after the Daily Scrum for detailed discussions, or to adapt, or re-plan, the rest of the Sprint's work.

The Daily Scrum is an internal meeting for the Development Team. If others are present, the Scrum Master ensures that they do not disrupt the meeting.

Daily Scrums improve communications, eliminate other meetings, identify impediments to development for removal, highlight and promote quick decision-making, and improve the Development Team's level of knowledge. This is a key inspect and adapt meeting.

Sprint Review

A Sprint Review is informal meeting, not a status meeting, held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed.



During the Sprint Review,

- The Product Owner explains what Product Backlog items have been "Done" and what has not been "Done";

- The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved;
- The Development Team demonstrates the work that it has "Done" and answers questions about the Increment;
- The Product Owner discusses the Product Backlog as it stands. He or she projects likely target and delivery dates based on progress to date (if needed);
- The entire group collaborates on what to do next;
- Review of how the marketplace or potential use of the product might have changed what is the most valuable thing to do next; and,
- Review of the timeline, budget, potential capabilities, and marketplace for the next anticipated releases of functionality or capability of the product.

The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities.

Remember to plan your Sprint Review. There are multiple parts to a Sprint Review (not just the demo). With a good plan, you can facilitate an amazing and effective review. Use the Sprint Event Planning Worksheet (in the Error! Reference source not found. **section beginning on page** Error! Bookmark not defined.**). Every Sprint Review should include:**

1. **The demo, including final decisions on DONE/NOT-DONE for each PBI.**
2. **The development team's evaluation of the Sprint, the challenges they faced with the product and how they solved them.**
3. **The product owner's evaluation of the Product Backlog regarding meeting expectations and deadlines.**
4. **The product owner's evaluation of the market, the intended use of the product, and whether or not the direction of the product should change.**
5. **The evaluation of the entire group in the review regarding the content and order of the Product Backlog.**

It's not JUST a demo!

Sprint Retrospectives

The Sprint Retrospective is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.

The purpose of the Sprint Retrospective is to:

- Inspect how the last Sprint went with regards to people, relationships, process, and tools;
- Identify and order the major items that went well and potential improvements; and,

- Create a plan for implementing improvements to the way the Scrum Team does its work.

The Scrum Master encourages the Scrum Team to improve, within the Scrum process framework, its development process and practices to make it more effective and enjoyable for the next Sprint. During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by improving work processes or adapting the DONEness Definition if appropriate and not in conflict with product or organizational standards.

No matter how you facilitate your Sprint Retrospective, help your team ensure that one or two action plans for improving the team's outcomes are defined and planned into the next Sprint.

Remember, if your team is moving forward in terms of skills and capabilities, they are moving backward.

By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint.

There are many ways to run a retrospective. Since the whole idea is to create an improvement plan for the team, no approach is ineffective if the team is engaged and talking. Common methods to approach a Sprint Retrospective are to discuss:

- What worked and what didn't work?
- What do we need to 1) start doing, 2) stop doing, 3) do more of, or 4) do less of?
- What's propelling us forward and what's holding us back?
- What made you happy during the Sprint and what made you angry?
- If you could improve anything about the previous Sprint, what would it be?

One important note: don't try to boil the ocean. Work with the team to identify one or two things that they want to improve and focus on those.

Quiz!

Please complete the following questions:

1. **What is the longest amount of time a Sprint may take?**

Please complete the following questions:
2. If a Product Owner asks for minor changes during a Sprint what are the possible options for handling the change?
3. Does my Development Team have to use the three questions in the Daily Scrum as defined by the Scrum Guide?
4. What are the outputs of the Sprint Planning Event?
5. What is the output of the Sprint Review Event?
6. Who owns the Sprint Backlog?

Please complete the following questions:
7. Which Scrum event occurs immediately when a Sprint is cancelled?
8. Bonus Question: Why would a Sprint be cancelled?

Conclusion

The key to creating a high performing team has NOTHING to do with productivity.

The key to creating a high performing team is totally about

CAPABILITY

Improve a team's capabilities, and productivity takes care of itself.

This is the end of the Basic Agile and Scrum Fundamentals training materials. Congratulations!

We recommend that you read the Scrum Guide (https://www.scrumguides.org). Our materials were built from that guide plus our own experiences. Check your notes as you go. Adding to your notes from the Scrum Guide will improve your recall for when you need the information.

If you have any questions, you can reach the Artisan Training Coordination Team at

training@artisanagility.com

Stay Connected with Artisan Agility

Your learning journey with Artisan doesn't need to end today. We hope it does not! Stay connected with us through

Detailed information about our training programs is located on our website; you can visit us at www.artisanagility.com.

We are active in the social media world, so you can

- Like us and engage with us on Facebook®
- Follow us on Twitter®
- Connect with us through LinkedIn®
- Share with us on Instagram®
- Subscribe to "Jim's Blog" the official Artisan Agility blog.

Quiz Answers

Page 8

1-Customers and stakeholders are encouraged to work directly with Scrum teams rather than relying on documentation as a guide.

2-Scrum is a very simple framework with three roles, four events, and only a few required artifacts. Teams are encouraged to experiment and improve.

3-The content and order of the Product Backlog (the list of work to be done on the product) can be modified at any time (with the exception that the content of an active Sprint is not affected).

Page 12

1-Scrum is very simple and, therefore, relatively easy to learn.

2-The Sprint

3-Self-organizing teams are generally higher-performing and create better outcomes. When given direction by an outside authority, they surrender all self-organization and become "hands" doing what they are told.

4-Teams that are "coder-only" (i.e., lack many of the skills necessary to do the job properly) will likely deliver poor quality and poor outcomes to the customer.

Page 16

1-A defined workflow is a repetitive process. The same inputs and the same outputs are expected. An empirical workflow allows us to use our skills, but apply them to a variety of different problems and solutions.

2-Commitment, courage, focus, openness, and respect.

3-transparency, inspection, and adaptation.

4-a working product increment is expected at the end of every Sprint; this makes the product "transparent" and allows us to inspect it and adapt our future plans appropriately.

5-Teams frequently decide to stop doing Sprint Retrospectives because the retrospectives aren't providing enough value. The most likely result of not doing Sprint Retrospectives is that the team will continue to make the same mistakes repeatedly.

Page 21

1-Product Owner

2-Product Owner

3-Development Team

4-Product Owner

5-Scrum Master

Page 31

1-Scrum Master

2-Conceptualization

3-Development Team

4-Nine

5-Active Listening

6-They are attempting to get the person to whom they are listening to identify a solution on their own. In this manner, the person OWNs the solution and is mostly likely to follow through.

Page 38

1-Eight hours

2-Development Team

3-Development Team

4-Product Owner and Development Team

5-Scrum Master

6-Product Owner

7-The development team member can say yes if the change can be made without threatening the Sprint Goal. Otherwise, they should say no. The Product Owner can then add the requested change to the Product Backlog if they wish.

Page 48

1-Sprint Backlog

2-Technical debt costs more to correct the longer it exists.

3-Size (estimate)

4-Product Owner

5-All pre-existing functionality still works, all new functionality works, there is likely more value that can be added, and the current increment may not yet be valuable enough to ship to the customer.

Page 57

1-One calendar Month

2-If it can be done without threatening the Sprint Goal, the development team can choose to do it; otherwise the Product Owner must decide to add it to the Product Backlog for later or forget it entirely.

3-No; the development team determines the proper format for the Daily Scrum, if the goals of keeping the development teams' members informed AND confirming that the Sprint Goal can still be achieved this Sprint is satisfied.

4-Sprint Goal and Sprint Backlog

5-A modified Product Backlog, ready for the next Sprint Planning event.

6-Development Team

7-Sprint Review

8-The company has changed its strategic direction OR market conditions change, making the content of the Sprint value-less.

Works Cited

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas. (2001). *Manifesto for Agile Software Development*. Retrieved from Manifesto for Agile Software Development: https://agilemanifesto.org

kanomodel.com. (n.d.). *What is the Kano Model?* Retrieved from www.kanomodel.com: https://www.kanomodel.com

Mark, G., Gudith, D., & Klocke, U. (2008). *The Cost of Interrupted Work: More Speed and Stress.* Irvine: Department of Informatics, University of California - Irvine.

Overeem, B. (2016, April 7). *The Scrum Master as an Impediment Remover*. Retrieved from Scrum.org: https://www.scrum.org/resources/blog/scrum-master-impediment-remover

Perlow, L. A., Hadley, C. N., & Eun, E. (2017, July). *Stop the Meeting Madness*. Retrieved from Harvard Business Review: https://hbr.org/2017/07/stop-the-meeting-madness

Sami, M. (2017, September 23). *Mohamed Sami*. Retrieved from Personal website – Software Engineering & Architecture Practices: https://melsatar.blog/2017/09/23/trade-off-analysis-technique-make-the-decision-easier/