3 Scrum Overview

SCRUM HAS BECOME RAPIDLY ACCEPTED as the most widely used agile methodology in the United States and its usage is rapidly expanding in other areas of the world as well. It provides a good general foundation that can be adapted to fit a very broad range of projects. Scrum is also not limited to software development, but that is where it is most widely used at the current time. For that reason, much of the discussion in this book will be focused on software development, but it should be understood that agile and Scrum are not limited to software development. For example, I used a Scrum approach to plan and organize the writing of this book.

Scrum, is, by definition, an *empirical process* as opposed to a "defined and predictive process." The following is how these two types of processes are different:

1. *Empirical process*. The empirical process control model was defined to exercise or control the process via following some frequent adaptations as well as frequent inspections. It works best in situations with high levels of uncertainty where it is difficult, if not impossible, to clearly define the solution in advance and an experimental, trial-and-error approach is needed to converge on an acceptable solution. The term *empirical process* control model is based on the word *empirical,* which means the information is acquired by the means of experimentation and observation.[1]
2. *Defined and predictive process*. "The defined process control model can be thought of as a theoretical approach. When a well-defined set of inputs is given, it is obvious that the same outcomes will be generated every time the program executes. With the well-understood technologies and stable requirements, one can very well predict a whole software project." [2]

In other words, a defined and predictive process model is appropriate if both the requirements for the project and the solution to those requirements can be accurately predicted in advance; however,

[1]Blogspot, "Explain Empirical versus Defined and Predictive Process?" April 5, 2012, http://productdevelop.blogspot.com/2012/04/explain-empirical-vs-defined.html
[2]Ibid.

that is typically not the case in a software development project. That is why an empirical process like Scrum is typically so much more successful in any environment such as software development with high levels of uncertainty.

Scrum is adaptive in two ways:

1. *Solution adaptation*. Scrum is adaptive in the sense of progressively defining the solution. You can start a Scrum project with only a very fuzzy idea of the requirements and the solution and progressively elaborate the requirements and the solution as the project proceeds to ultimately converge on a solution that meets the business need.
2. *Process adaptation*. The process itself behind Scrum is also adaptive. A Scrum project is broken up into short, fixed-length sprints. At the end of each sprint as the project proceeds, the project team pauses to determine if the process is working or needs to be further adapted to fit the nature of the problem.

Ken Schwaber, one of the original developers of Scrum, provides this overview of Scrum:

> I offer you Scrum, a most perplexing and paradoxical process for managing complex projects. On one hand, Scrum is disarmingly simple. The process, its practices, its artifacts, and its rules are few, straightforward, and easy to learn . . . On the other hand, Scrum's simplicity can be deceptive. Scrum is not a prescriptive process; it doesn't describe what to do in every circumstance. Scrum is used for complex work in which it is impossible to predict everything that will occur. Accordingly, Scrum simply offers a framework and a set of practices that keep everything visible. This allows Scrum's practitioners to know exactly what's going on and to make on-the-spot adjustments to keep the project moving toward desired goals.
>
> Common sense is a combination of experience, training, humility, wit, and intelligence. People employing Scrum apply common sense every time they find the work is veering off the path leading to the desired results. Yet most of us are so used to prescriptive processes—those that say 'do this, then do that, and then do this'—that we have learned to disregard our common sense and instead await instructions.[3]

SCRUM ROLES

Scrum is based on some very well-defined roles that are discussed in this section.

[3] Ken Schwaber, *Agile Project Management with Scrum* (Redmond, WA: Microsoft Press, 2004), p. 1.

Product owner role

The *product owner* role in Scrum is defined as follows:

> The Product Owner is responsible for maximizing the value of the product and the work of the Development Team. How this is done will vary widely across organizations, Scrum Teams, and individuals.
>
> The Product Owner is the sole person responsible for managing the Product Backlog. Product Backlog management includes:
>
> - Clearly expressing the Product Backlog items;
> - Ordering the items in the Product Backlog to achieve goals and missions;
> - Optimizing the value of the work the Development Team performs;
> - Ensuring that the Product Backlog is visible, transparent, 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.[4]

The Product Owner represents the business sponsor, is a decision maker for the project, provides direction to the team on what will be built, and prioritizes the work to be done. On a typical Scrum project, there is only one product owner; however, on large enterprise-level projects, product owners might not act alone. They might gather inputs from multiple stakeholders and, in some cases, need to coordinate with product owners of related efforts.

Relationship to Project Management Role

The product owner role has many elements of functions that might be considered project management in providing direction to the project but it goes beyond a typical project management role and includes the domain knowledge to provide the business direction to a project as well. This role is very similar to a product manager in a product development company. The product owner has some responsibilities that might be similar to a project manager, but it is a very different role. In a traditional project, a project manager doesn't define the requirements of what should be developed and that is what a product owner does.

[4]Ken Schwaber and Jeff Sutherland, "Scrum Guide" Scrum.org (July 2013), https://www.scrum.org/scrum-guide.

He has some of the attributes of a business analyst in collecting and defining requirements but the role goes well-beyond the role that a business analyst would be expected to play in being a decision maker on the product features and capabilities.

Scrum Master role

The Scrum Master role in Scrum is defined as follows:[5]

> The Scrum Master is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules.
>
> 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 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 Scrum Master serves the Product Owner in several ways, including:
>
> - 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 and needed.
>
> The Scrum Master serves the Development Team in several ways, including:
>
> - 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;
> - Facilitating Scrum events as requested or needed; and,

[5] Ibid.

- Coaching the Development Team in organizational environments in which Scrum is not yet fully adopted and understood.

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

- Leading and coaching the organization in its Scrum adoption;
- Planning Scrum implementation 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.

The Scrum Master is what is known in agile as a *servant leader*. He/she has the responsibility to facilitate the team. He/she is not expected to provide a significant amount of direction to the team as a *project manager* would—the team is supposed to be self-organizing and empowered so the role of the Scrum Master in providing direction is limited. However, in the real world, many teams are not at a level of maturity to act effectively as a self-organizing and empowered team, and the Scrum Master might have to be somewhat of an agile coach to help the team reach that level of maturity.

Relationship to Project Management Role

Some of the responsibilities of the Scrum Master might be considered similar to a project manager. For example, the Scrum Master is expected to track and remove obstacles that are inhibiting the performance of the team. However, it is also a very different role—the Scrum Master is expected only to facilitate the team without providing much direction. There are a couple of different views on this:

- The "idealistic" view is that the team is totally empowered and self-organizing, and very little or no direction of any kind is required.
- A more pragmatic view is that not all teams are at that level of maturity, and some level of leadership is needed. The Scrum Master has to really "fill the cracks" to fit the situation.

The right solution, in my experience, has been to take an adaptive approach to provide whatever leadership is needed to help the team be successful; however, a very important key point is that this is not a traditional project management role. At the team level, there is no defined role for a project manager in a Scrum project.

Team role

The role of the team in an agile project is defined as follows:[6]

> 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. 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 of the skills as a team necessary to create a product increment;
> - Scrum recognizes no titles for Development Team members other than Developer, regardless of the work being performed by the person; there are no exceptions to this rule;
> - Scrum recognizes no sub-teams in the Development Team, regardless of particular domains that need to be addressed like testing or business analysis; there are no exceptions to this rule; and,
> - Individual Development Team members may have specialized skills and areas of focus but accountability belongs to the Development Team as a whole.

The team is expected to act as a single entity where all the members of the team act collaboratively and cohesively as a single unit. A team may consist of people with different specialties. For example there could be people who specialize in development on the team and others who specialize in testing. It might also include business analysts within the team.

There are also different views regarding how a team is made up:

- The idealistic view is that everyone on the team is equally capable of performing any task required by the team.
- The more pragmatic view is that, in the real world, some specialization by skill within the team is beneficial. Having different skills within the team is not inconsistent with having good teamwork,

[6] Ibid.

in my opinion. In a sports team, you have people with very different skills and abilities who play different positions on the team that are well aligned with their unique skills and abilities. Those are very different skills, but they can still work together as a well-coordinated, cohesive team. In fact, an argument can easily be made that the team should be cross-functional and having different perspectives within the team is more effective in many circumstances than having a team of people who all think absolutely alike in all situations.

Relationship to Project Management Role

The team also has some responsibilities that might be considered similar to a project manager. Each member of the team is expected to plan their activities and be accountable for delivering the results that they committed to. In a sense, agile distributes responsibility downward into the team.

Scrum framework

Scrum is generally called a *framework* rather than a *methodology* because it is meant to provide a framework for organizing the work rather than a more specific, well-defined methodology or how the work should be done. A high-level overview of the Scrum framework is shown in Figure 3.1.

Each of the elements of the framework is discussed in more details in the following sections.

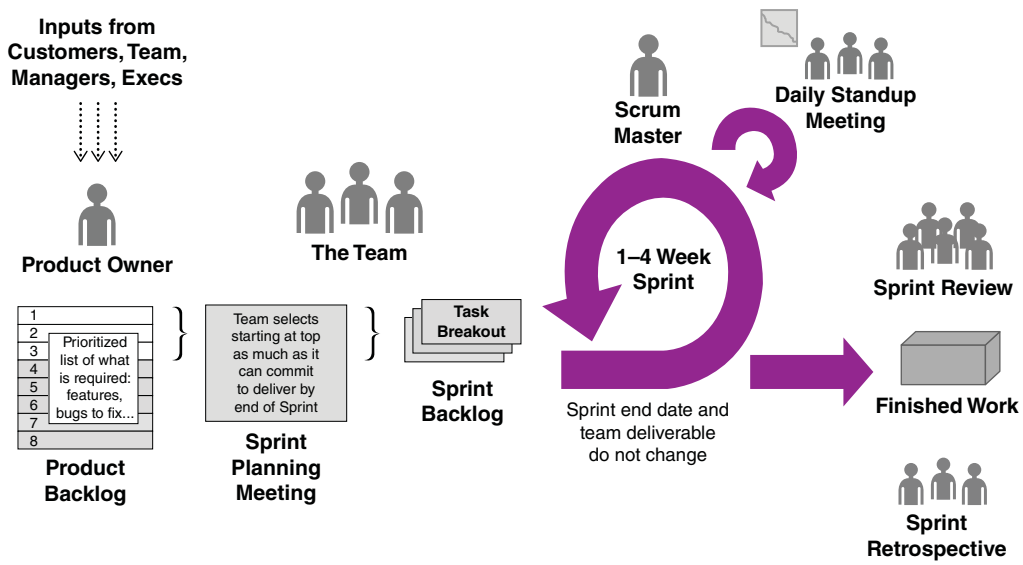

FIGURE 3.1 Scrum framework

Courtesy of Rally

Sprints

The heart of the Scrum process is the *sprint*. A "sprint is typically a fixed-length time-box and is generally from two to four weeks in duration and is where the development work is actually done. Instead of a traditional *schedule-boxing* approach in which the length of a project or an iteration is expanded as needed to be long enough for whatever is being developed, the length of a sprint is fixed and the work to be done is broken up into small increments where no single increment is so big that it cannot be accomplished in a single sprint. Instead of expanding the length of the sprint to fit the work to be done, the challenge is to see how many of these small increments of work can be fit into a single fixed-length sprint.

Product Backlog

The product backlog is a queue of the work to be done organized in the form of small increments of work typically in the form of *user stories*. (User stories will be discussed in more detail later; however, they are a very streamlined way of defining a very small and individual user requirement.) A good practice with *user stories* is to describe the user requirement in very concise terms and also briefly define what the user expects to accomplish with the user story as an acceptance criteria. The product backlog is dynamic and is continuously prioritized by the product owner to order the items in the product backlog to deliver the most business value as early as possible. As new ideas come up during the course of the project, they will also be added to the product backlog so that the product backlog continuously expands and contracts as work is completed and new work is added and is continuously reviewed, approved, and prioritized by the product owner.

The product backlog is also used to hold all work that is to be completed, including any defects that are deferred from the current sprint. Normally, defects would be resolved in the current sprint, but the product owner can make a decision to defer that work if necessary. However, building up an excessive amount of defects in the product backlog is referred to a *technical debt* and should not be allowed to grow out of control.

The product owner in Scrum is responsible for defining and managing the product backlog on an ongoing basis throughout the project; however, in many situations he/she is assisted in that role by the Scrum Master. If the project warrants it, in some cases, a business analyst may also assist the product owner with managing the product backlog.

There are different ways that the product backlog might be managed, depending on the nature of the project—in a very adaptive project approach with uncertain requirements, the product backlog might be limited to only looking two to three sprints into the future. However, that approach would not provide much of an ability to plan the schedule and costs of a project. If there is a need for longer term planning, naturally, the product backlog would need to be defined further out in time; however, it could be limited to a high-level definition only for items that are very far out in time. The product backlog will be discussed in more detail in a later section.

A very important step in the Scrum process that is not explicitly shown in this diagram is product backlog grooming. In product backlog grooming, the team (or part of the team including the product

owner) meets regularly to "groom the product backlog," in a formal or informal meeting which can lead to any of the following:[7]

- Removing user stories that no longer appear relevant
- Creating new user stories in response to newly discovered needs
- Reassessing the relative priority of stories
- Assigning estimates to stories which have yet to receive one
- Correcting estimates in light of newly discovered information
- Splitting user stories that are high priority but too coarse-grained to fit in an upcoming iteration

The product owner is responsible for product backlog grooming; the Scrum Master facilitates it; expert "leads" (SMEs, developers, QA) sometimes need to opine; and stakeholders need to contribute.

The review and refinement (especially business prioritization and sizing) of the product backlog is done in parallel *as the team is building* during a sprint. If product backlog grooming is not done, the next sprint planning will not have a good product backlog ready for the product owner and team to agree on what the next sprint needs to include. Instead, they may have to spend a day or two doing the evaluation, which could delay the start of the next sprint.

Sprint planning

Sprint planning plays a very important role in Scrum. The sprint planning meeting takes place prior to the beginning of every sprint. It has two major goals and is typically broken into two parts to align with those two goals:

1. The goal of the first part is for the product owner and the team to negotiate what stories will be taken into the sprint.
   - The product owner makes any necessary final revisions in the priority ranking of the potential stories to be taken into the sprint.
   - The team decides how many of those stories can be completed in the sprint based on the level of effort required for each story and the estimated team capacity or velocity.

Note

> "Velocity is the measure of the throughput of an agile team per iteration. Since a user story, or story, represents something of value to the customer, velocity is actually the rate at which a team delivers value. More specifically, it is the number of estimated units (typically in story points) a team delivers in an iteration of a given length and in accordance with a given definition of 'done.'"[8]

[7] "Backlog Grooming," Agile Alliance, http://guide.agilealliance.org/guide/backlog-grooming.html.
[8] "Agile Sherpa–Velocity," VersionOne, http://www.agilesherpa.org/agile_coach/metrics/velocity/.

This part of the planning meeting is essentially a negotiation between the product owner and the team.

2. The goal of the second part is for the team to define the tasks that will be needed to implement those stories and to plan how those stories and tasks will be allocated among the members of the team. The product owner typically does not participate in this portion of the sprint planning meeting.

   The product backlog should be groomed and prioritized by the product owner prior to the sprint planning session; however, final adjustments in the priority ranking are likely to take place in the sprint planning meeting as the team and the product owner negotiate and plan what can be taken into the sprint and resolve any trade-offs.

The sprint planning meeting is typically time-boxed to four hours. The largest part of that time is typically spent in the second part of task-level planning by the team. During the sprint planning meeting, the product owner and the team should agree on an overall goal to be accomplished in the sprint in addition to negotiating the stories that will be taken into the sprint. Once that is done, the team typically does the task-level planning portion of the meeting without the product owner.

Daily standup

The Daily Standup is basically a check-in for everyone on the team to coordinate what's going on, monitor progress, and to identify any obstacles that may be inhibiting progress. During the Daily Standup, each person on the team typically answers three questions:

1. What did you accomplish yesterday?
2. What are you going to accomplish today?
3. What obstacles are in your way?

The meeting is facilitated by the Scrum Master and it is often done in front of a Scrum board that indicates the progress of the tasks in the current sprint. Figure 3.2 shows an example of a Scrum board. Alternatively, for distributed teams, an online agile project management tool is typically used in lieu of a physical Scrum board. (See Chapter 10 VersionOne Tool Overview.)

Usually, everyone on the team stands up during these meetings as an incentive to keep the meeting short and focused (that's why it's called a *Standup*). However, it is common to deviate from those rules with distributed teams. With distributed teams, the Daily Standup may be one of only a few opportunities for communication among the team, so it might be necessary to go beyond the three basic questions, but it is a good idea to still limit the length of the meeting and defer lengthy topics to other meetings.

Sprint review

The sprint review is where the team presents the finished work to the product owner for his/her final review and approval. It is essentially a brief User Acceptance Test at the end of each sprint. It should

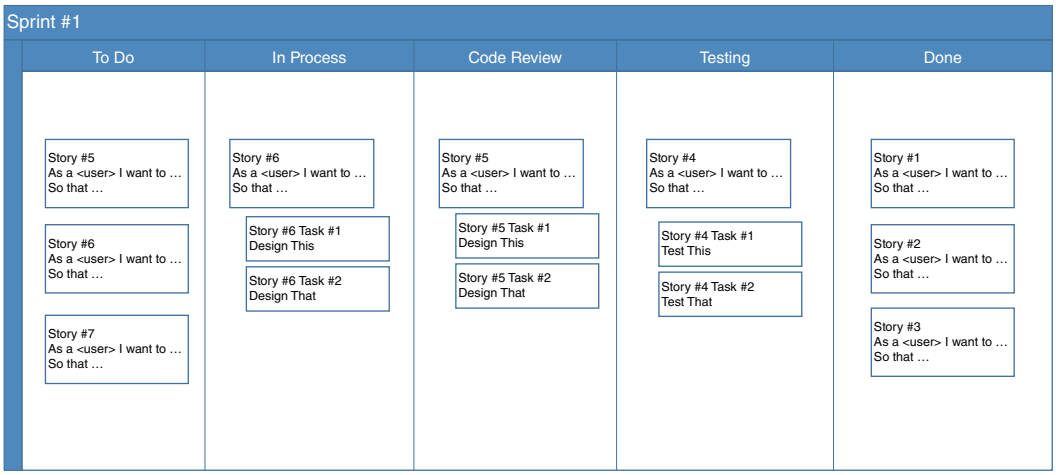

FIGURE 3.2 Example Scrum board

be noted that the sprint review should not be the very first time that the product owner has seen the results of the software that the team is developing. The team should demo and preview the results of their work to the product owner as it is being developed during the sprint to get feedback and inputs. It is also a good practice to not wait until the sprint review for final acceptance of each story. It's a good idea to present a story to the product owner for review as soon as it has been completed in the sprint. The sprint review is essentially a formal review of all the items in the sprint.

All defects in the product should typically be resolved prior to the sprint review unless the product owner has specifically approved deferring the resolution to a later sprint. If the product owner requests significant changes or enhancements in the stories, those will typically be added to the product backlog and deferred until a future sprint. Significant changes to stories that are in progress should not be allowed in the middle of a sprint—changes while the sprint is in progress should be limited to clarification and refinement of user stories only.

Although the product owner has primary responsibility for approving the results of the sprint review and should represent the interests of all business users, it is often a very good practice to include business users and other stakeholders who might have input into the sprint review in these meetings.

Sprint retrospective

The sprint retrospective is an opportunity to review and discuss lessons learned at the end of the sprint where the team looks back, reflects on what went well and what didn't go well, and identifies opportunities for process improvement in the next sprint. This is a very important element of the Scrum process. A sprint retrospective is to a sprint in an agile project as a post mortem is to a project

in a traditional project. It is an opportunity to stop and reflect on the process and make improvements. Because it is done at the end of each sprint and sprints take place every two to four weeks, learning and process improvement can be much more rapid than a traditional project. An agile development process is not intended to be rigid; it is intended to rely heavily on continuous improvement to adapt the process to the project and the environment as much as possible. Because sprints are short, learning can be very fast so that mistakes and problems are not allowed to propagate very far.

GENERAL SCRUM/AGILE PRINCIPLES

It's important to not get lost in the mechanics of Scrum and to understand the principles behind it in order to apply the methodology intelligently in a variety of different project and business environments.

> People tend to interpret Scrum within the context of their current project management methodologies. They apply Scrum rules and practices without fully understanding the underlying principles of self-organization, emergence, and visibility and the inspection/adaptation cycle. They don't understand that Scrum involves a paradigm shift from control to empowerment, from contracts to collaboration, and from documentation to code.[9]

Kenneth Rubin has provided a very nice summary of the important principles behind Scrum in his book, Essential Scrum: A Practical Guide to the Most Popular Agile Process[10] that are further discussed here. These principles are really not unique to Scrum; they are general enough to apply, to some extent, to almost any project management approach (either plan-driven or adaptive). They just have to be used intelligently in the right proportions based on the nature of the project.

Variability and uncertainty

An understanding of variability and uncertainty and how to manage them effectively is probably one of the most critical requirements for any agile project manager to master. Kenneth Rubin identifies several important principles associated with managing variability and uncertainty:[11]

- *Embrace helpful variability*. Plan-driven processes are built around controlling and limiting change. In a plan-driven process, any change is considered an exception to the norm and must be controlled in order to manage the scope, costs, and schedule of the project. That can lead to a very

[9]Schwaber, *Agile Project Management with Scrum,* p. 26.
[10]Kenneth Rubin, *Essential Scrum: A Practical Guide to the Most Popular Agile Process* (Upper Saddle River, NJ: Pearson Education, 2013).
[11]Ibid., p. 32.

inflexible process where you may wind up meeting cost and schedule goals but fail to provide the desired business value because the process wasn't adaptive enough to meet customer needs. This is probably the primary reason agile has been so successful in a very dynamic and rapidly changing business environment.

An agile process is based around easily adapting to customer needs and embracing change; however, change in an agile process is not totally unrestricted and a sensible approach is still needed to manage changes.

- *Employ iterative and incremental development*. Agile is heavily based on an incremental development; instead of trying to build the entire product all at once, the product is broken up into smaller pieces and built incrementally with an opportunity for feedback and learning after each increment is completed.
- *Leverage variability through inspection, adaptation, and transparency*. Scrum is heavily based on inspection and adaptation. Not only is the product being built continuously and inspected and adapted as needed to maximize the value it provides to the user, but the process itself used to build the product is also continuously inspected to determine if the process is working optimally and any adjustments are made to the process as necessary. Transparency about both the product and how the process works is essential to accomplish that.
- *Reduce all forms of uncertainty simultaneously*. In any project there are a variety of kinds of uncertainty:
  - End uncertainty deals with uncertainty associated with the features of the final product.
  - Means uncertainty surrounds the process and technologies used to develop the product.
  - Customer uncertainty relates to who the ultimate customer of the product will be and what their needs are.

  A plan-driven project attempts to remove all uncertainty about the product and how it will be built up front, and that's just not realistic in many situations. An agile project is built around recognizing, managing, and reducing uncertainty as the project progresses.

Prediction and adaptation

Many people have tried to implement Scrum dogmatically and rigidly "by the book," when it was intended to be adapted to the nature of the project. An important set of principles behind Scrum identified by Kenneth Rubin is related to prediction and adaptation. The following are the key elements of this principle that he has identified:[12]

- *Keep options open*. Plan-driven approaches frequently try to make all decisions about the requirements upfront to resolve any uncertainty about the project before it starts. Scrum and agile are

[12]Ibid., pp. 37–39.

based on delaying decisions until "the last responsible moment". That is, in general, you should delay making decisions as long as they can be delayed without impacting the project. The reasoning behind that is that if you delay making decisions, you will typically have better information for making those decisions and ultimately make better decisions. If you try to make decisions too far in advance it will frequently require speculation and many times that speculation will be wrong and the effort may be wasted because that decision will only need to be re-planned later when better information is available.

- *Accept that you can't get it right up front*. An important principle in Scrum and agile is to accept that some amount of trial-and-error experimentation may be necessary to find the best solution. This requires a mature attitude to recognize that we "don't know what we don't know."
- *Favor an adaptive, exploratory approach*. There is an economic decision associated with weighing the cost of experimenting and adapting to the results of the experimentation versus the cost of designing the product up front and the risk of major rework and redesign if it is wrong. In the 1990s, the cost of making changes was significant, which is one of the reasons that led to developing heavily plan-driven approaches. However, tools and technologies have gotten a lot better since that time, and the cost of exploratory development has come down significantly.
- *Embrace change in an economically sensible way*. Agile and Scrum provide sensible ways to manage change to minimize the economic impact to the project of those changes. With traditional plan-driven approaches, the cost of changes can rise significantly later in the project because the impact can be significant. By using an incremental approach to development in an agile project, the impact of changes can be limited and the costs of making changes can be relatively flat throughout the project.
- Balanc*e predictive up-front work with adaptive just-in-time work*. In any project, there is always a certain amount of information that is known or can be easily known upfront and some that is much more difficult to determine upfront. It would be foolish to ignore what is already known. A sensible approach is to balance a predictive up-front approach with an adaptive just-in-time approach to resolve uncertainties based on the nature of the project.

Validated learning

A third area of principles that Kenneth Rubin has identified is related to "Validated Learning."[13] As we've mentioned, agile is based heavily on an empirical and experimental approach to try things and see what works. To make that approach work, it is important to recognize that we don't have all the answers and we might have to make some assumptions, but it is important to recognize that those assumptions are only assumptions and need to be validated.

[13]Ibid., pp. 44–46.

- *Validate important assumptions fast*. If there are specific assumptions or decisions that might have a significant impact on a number of areas of the project that might require significant rework if they're not made correctly, it makes good sense to identify those areas and resolve them as quickly as possible to minimize that impact.
- *Leverage multiple concurrent learning loops*. There are multiple concurrent learning loops in a Scrum project that operate at different levels, and it is best to take advantage of all of these different levels concurrently. For example, there is a level of learning that takes place in the daily Scrum meetings and there is also a level of learning that takes place at the end of each sprint in the sprint review and sprint retrospective. We should take advantage of all these sources of learning.
- *Organize workflow for fast feedback*. We should organize the workflow in the project so that we get feedback as rapidly as possible on the items where feedback is most essential to resolve uncertainties and validate assumptions.

Work in progress

The next group of principles identified by Kenneth Rubin is related to work in progress, or WIP. The important elements of the principles that he has identified in this area are:[14]

- *Use economically sensible batch sizes*. Traditional, plan-driven development processes have been based on the principle of economy of scale that comes from a manufacturing environment. In a manufacturing environment, it may make sense to perform a repetitive manufacturing operation on large batch sizes to reduce costs, but there are very different economies of scale associated with a product development process and small batch sizes tend to be more efficient in that environment for a number of reasons:
  - Work can be distributed more evenly and flow through the process is maximized where large batch sizes can cause serious bottlenecks and reduce overall flow.
  - Faster cycle time means feedback is quicker, which results in faster learning and adaptation.
  - Risk is also reduced.
- *Recognize inventory and manage it for good flow*. Manufacturing plants are very much aware of the cost and impact of carrying a large amount of inventory. In a software development process, "inventory" is not as visible, and the impact of carrying inventory is not as well understood. Inventory in a product development process consists of a variety of different kinds of items of work in process including requirements, code, and so on. There is always a cost and a certain amount of overhead associated with managing those items of inventory, so it is best to keep work in process inventory as small as possible to make the overall process more efficient.

[14] Ibid., pp. 48–52.

- *Focus on idle work, not idle workers*. There are two kinds of waste that are important to recognize in a project:
  - One is *idle worker waste* where workers are not being fully utilized at 100 percent of their capacity and some component of their capacity is not being used and wasted that might be better utilized.
  - The other is *idle work waste*, which is when work sits idle until people are available to work on it.

    Both of these forms of waste are important, but traditional, plan-driven projects are typically very heavily focused on *idle worker waste* and attempt to make sure that everyone in the project is working at 100 percent of capacity. An agile project recognizes that both of these forms of waste are important and *idle work waste* is at least equally important if you want to optimize the flow of work through the process.
- *Consider cost of delay*. A related consideration is the cost of delay. If a project is delayed waiting for resources to be available, the impact on the cost of the project might be significant because everyone on the project might be delayed for some period of time. The example that Kenneth Rubin gives is that delaying the completion of documentation until the end of the project might delay the entire project by two to three months, and there could be a significant cost in that delay.

Progress

Another set of principles identified by Kenneth Rubin is related to progress. He has identified the elements of these principles as follows:[15]

- *Adapt to real-time information and replan*. Traditional projects are based on conformance to plan, which overlooks the fact that the plan could be wrong or the plan might need to be adapted to changing conditions as the project progresses. An agile/Scrum project is much more adaptive and is based on the assumption that the plan will be continuously revised based on learning and information that emerges throughout the process.
- *Measure progress by validating working assets*. Traditional projects often measure progress by estimating percent complete of the tasks required to complete the project. That is typically based on a very subjective judgment that can be very inaccurate.

  A traditional project also has some major risks to progress that aren't fully considered in evaluating progress. For example, a task might be complete but if it isn't tested and accepted by the customer, is that really complete? A much better way to measure progress is measuring completed working assets that have been validated and accepted by the customer as they are being built.

[15]Ibid., pp. 54–55.

- *Focus on value-centric delivery*. Traditional, plan-driven projects typically perform final integration, testing, and delivery of all features at the end of the project. With this approach, there is a risk that the project will run out of time and money before delivering all of the important value to the customer. Also, we all know that there are many traditional, plan-driven projects that are bloated and/or "gold-plated" where a large percentage of the features are not really essential. This often results from the customer's belief that if they don't get some item that they might want into the requirements up front, they may never get it at all.

  Agile works by prioritizing the features to be delivered based on value and developing and by delivering those features incrementally, we can avoid the "all or nothing" approach that typically occurs in traditional, plan-driven projects and at least deliver the most important value to the customer quickly. There have been many situations where after some percent of the value is delivered, the customer believes that his essential needs are satisfied, there may be diminishing returns associated with completing the rest of the functionality, and there may be no need to complete the remainder of the project. This can result in a considerable savings in development costs.

Performance

The final set of principles identified by Kenneth Rubin is related to performance. He has identified the elements of these principles as follows:[16]

- *Go fast but never hurry*. An agile project is based on being nimble, adaptive, and speedy; however, rushing too quickly to get things done may not be the best approach and may have undesirable side effects such as burning out the people who work on the project and perhaps even compromising the quality of the product.
- *Build in quality*. In a traditional, plan-driven project, testing might typically be done in a later phase so the approach to quality might be somewhat reactive and focused on finding and fixing defects well after the development has been completed. There are a number of very significant potential problems associated with that approach:
  - Deferring finding problems until very late in the project, where it might be much more difficult to resolve the problems, can make detecting and resolving individual defects much more difficult. When problems are allowed to accumulate without being resolved, there is a compounding effect that can make it very difficult to isolate and resolve individual problems. It's like trying to untangle a gigantic ball of twine with lots of knots in it.

    As an example, I worked at a large electronics company in the early 1990s that spent over $150 million developing a large, complex communications switching system. Much of the testing to integrate all of the components was deferred until the very end, and because it was such

[16] Ibid., pp. 57–58.

a complex system, it became very difficult to isolate individual problems. The whole project was canceled, and the company had to default on delivery commitments to several very large and important beta test customers.

- There is also a large risk that problems won't be found at all if testing is not directly associated with incremental functionality as it is developed and testing is delegated to an independent group that doesn't have primary responsibility for developing the product. In that situation, the people doing the testing may not be fully aware of the functionality that has been developed; and, as a result, the testing may be incomplete.

  Agile is heavily based on the principles in TQM, which emphasizes a more proactive process to eliminate defects at the source rather than finding and fixing them later. Manufacturing environments learned that lesson a long time ago. Years ago, there used to be a lot of quality control inspectors at the end of an assembly line that would inspect products and reject any that were defective. That was a costly process, not only in terms of products that had to be discarded or reworked but also in the cost of the many inspectors it took to find those defects. Also, any inspection approach like that is based on sampling, and it would be prohibitively expensive to inspect 100 percent of the product so some defects are bound to get through undetected.

  Going upstream in the process and designing the process to be inherently reliable in producing products that were free of defects, companies were able to significantly reduce the cost of QC inspectors and produce much more reliable products. Also, putting the responsibility on the people doing the development of the product to produce quality products that are free of defects rather than relying on an independent group to try to find defects later in the process has a very significant impact.

- *Employ minimally sufficient ceremony*. Traditional, plan-driven projects tend to be high ceremony, document-centric, process-heavy approaches. Scrum and agile use a much leaner process where ceremony, documentation, and process are limited as much as possible to only items that contribute value to the project.

  A good example is documentation. There is a common misconception that an agile project does not produce any documentation. That is not the case but documentation should not be an end-in-itself. We shouldn't produce documentation for the sake of documentation. If a document serves an important purpose and provides value in some way, it should be included in the project, but there are many ways to simplify documentation to make it satisfy the purpose it was intended for more efficiently.

  Here's an example of a streamlined approach to documentation: instead of writing detailed functional specs to define a set of features, some simple user stories can be created, and some very succinct acceptance tests can be defined in conjunction with the user stories to better define the requirements that story must satisfy. Those very simple user stories and very succinct acceptance criteria can eliminate the need for highly detailed functional specifications and, in most situations, provide a more effective solution.

SCRUM VALUES

Having a consistent set of values among everyone who participates in a Scrum team plays a critical role in helping to develop a collaborative, cross-functional approach. When people act from common values, it helps to unify everyone on the team, reduces potential conflict, and provides a solid foundation for guiding the team. The following is a summary of the Scrum values.

Commitment and focus

The first Scrum value is commitment and focus. As stated in the Scrum Alliance Code of Ethics:

> Commitment is our willingness to dedicate ourselves to a goal and to do our best to meet that goal. Focus means that we concentrate on and are answerable for doing the things that we have committed ourselves to do, rather than allowing ourselves to become distracted or diverted.
>
> As practitioners in the global Scrum community:
>
> - We take responsibility for and fulfill the commitments that we undertake—we do what we say we will do.
> - We make decisions and take actions based on the best interests of society, public safety, and the environment.
> - When we make errors or omissions, we take ownership and make corrections promptly. When we discover errors or omissions caused by others, we promptly communicate them to the appropriate individual or body. We accept accountability for any issues resulting from our errors or omissions and any resulting consequences.
> - We protect proprietary or confidential information that has been entrusted to us.
> - We proactively and fully disclose any real or potential conflicts of interest to the appropriate parties.[17]

Commitment is very important:

› The team needs to agree on a goal and the items that will be accomplished for each sprint during the sprint planning meeting and the team should hold itself accountable for successfully completing those items and goals as the sprint progresses.

[17] *Scrum Alliance Code of Ethics,* http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CB0QFjAA&url=http%3A%2F%2Fmembers.scrumalliance.org%2Fresource_download%2F1687&ei=tJTmU5OBL8-dygSO_YLADQ&usg=AFQjCNFUdwoEFFD_gqPoN4g2j7hzANCqdw&sig2=GdjGuSvPhIf_tYCLOsmXGg&bvm=bv.72676100,d.aWw.

› The idea of team accountability is a very important notion in Scrum. The whole idea of a self-organizing team relieves a lot of burden on the need for someone to manage the efforts of the team. If it works properly, there is no need for a project manager to provide direction to the team and track progress at the team level.

The idea of focus is also very important—once the items to be completed during the sprint have been agreed to, they should not be radically changed in the middle of the sprint. It is understood that details associated with items will be elaborated during the sprint; however, nothing should be allowed that changes the focus of the team during the sprint.

Openness

The second Scrum value is openness. Being able to be open with others is a sign of emotional maturity and it is essential for developing high performance teams that work collaboratively with each other. We've also seen immature people and teams who have hidden agendas and political goals that might supersede and conflict with the goals of the team.

Once people have worked on Scrum teams and have learned to practice these values, it becomes much easier to assemble a team who has shared values, but it can be difficult to get an immature team with no prior experience in Scrum to this level. The *Scrum Alliance Code of Ethics* says that as practitioners in the global community:

- We earnestly seek to understand the truth.
- We strive to create an environment in which others feel safe to tell the truth.
- We are truthful in our communications and in our conduct.
- We demonstrate transparency in our decision-making process.
- We provide accurate information in a highly visible and timely manner.
- We make commitments and promises, implied or explicit, in good faith.
- We do not engage in or condone behavior that is designed to deceive others.[18]

Teams typically go through stages of maturity in developing openness with each other. An excellent model of team dynamics is the *Tuckman model* that describes several different stages of team dynamics:[19]

1. Forming
    - No prior experience working with others on the team; team members may be somewhat reserved, formal and polite with each other'

[18]Ibid., p. 2.
[19]"Forming, Storming, Norming, and Performing," Mind Tools, http://www.mindtools.com/pages/article/newLDR_86.htm.

- Some members of the team may be apprehensive and unsure of themselves with the rest of the team.
- There is uncertainty about how the team will evolve.

2. Storming
   - People on the team begin to openly challenge each other.
   - Some conflict occurs and is essential to get past before the team can progress.
3. Norming
   - The team agrees to work together around some defined rules.
4. Performing
   - The team reaches the highest level of performance and works naturally and collaboratively and is able to dynamically adjust to new situations.

The theory is that teams need to go through these stages in order to make progress, but it isn't necessarily totally sequential. Sometimes teams will move back and forth between stages in order to make progress. For example, *storming* and conflict should be viewed as a sign of progress because it is essential for teams to go through that stage before making real progress towards the higher level of "performing" as a truly cross-functional and collaborative team based on openness.

Respect

Showing respect is a very important Scrum value. It is well known that high-performance teams may be composed of people with different opinions. If everyone on the team was a yes man and went along with everything others on the team said automatically, the team would probably not be very high performing. There is a lot of value in hearing different points of view and reaching consensus, and that calls for respect. The *Scrum Alliance Code of Ethics* states it this way:

> Respect means that we show a high regard for ourselves, others, and the resources entrusted to us. Resources entrusted to us may include people, money, reputation, the safety of others, and natural or environmental resources.
>
> An environment of respect engenders trust, confidence, and performance excellence by fostering mutual cooperation and collaboration—an environment where diverse perspectives and views are encouraged and valued.
>
> As practitioners in the global Scrum community:
>
> - We respect the rights and beliefs of others.
> - We listen to others' points of view, seeking to understand them.
> - We approach directly those persons with whom we have a conflict or disagreement.
> - We conduct ourselves in a professional manner, even when it is not reciprocated.

- We negotiate in good faith.
- We do not exercise the power of our expertise or position to influence inappropriately the decisions or actions of others in order to benefit personally at their expense.
- We do not discriminate against others based on, but not limited to, gender, race, age, religion, disability, nationality, or sexual orientation.
- We do not engage in any illegal behavior.[20]

Courage

The final Scrum value is courage. In my experience, this value becomes important in several different ways:

1. Scrum is difficult and challenging, and doing it successfully requires courage to face and overcome many obstacles.
2. It requires facing up to reality openly and honestly.
3. You need courage to overcome your own personal inhibitions to interact with others. The *Scrum Alliance Code of Ethics* states it this way:

> Courage means that we have the daring to do the best that we can and the endurance not to give up. We have the determination and resolution to take ownership of the decisions we make or fail to make, the actions we take or fail to take, and the consequences that result.
>
> As practitioners in the global Scrum community:
>
> - We share bad news even when it may be poorly received.
> - We avoid burying information or shifting blame to others when outcomes are negative.
> - We avoid taking credit for the achievements of others when outcomes are positive.
> - We would rather say, "No," than make false promises.
> - We accept the possibility of failure but also know that we can learn from failure and apply those learnings to our next attempt.
> - We acknowledge that change is inevitable, that change leads to growth, and that growth guides us toward improvement.
> - We admit when we need help and we ask for help.[21]

[20] *Scrum Alliance Code of Ethics*, p. 2.
[21] Ibid., p. 3.

SUMMARY OF KEY POINTS

1. Scrum Overview

   Scrum is based on an empirical process control model; it is adaptive at two levels (1) the definition of the solution is adaptive and (2) the process itself is adaptive.

2. Scrum Framework

   Many people think of agile as being very loosely defined, with a very minimally defined process—that is not really accurate. The Scrum process is actually very well defined and requires a good deal of skill and discipline. It is a different kind of discipline in that the process itself is intended to be very dynamic and adaptive.

3. General Scrum/Agile Principles

   It's very important to not get lost in the mechanics of how to do Scrum—having a defined methodology is important, but it is intended as a basis for ongoing continuous improvement. Both the process and the project deliverables are expected to evolve with the project based on rapid learning at the end of each sprint.

   It's also very important to understand the principles behind Scrum in order to do whatever adaptation may be necessary to apply it to unique projects and business environments.

4. Scrum Values

   Values in Scrum are important. Having a consistent set of values among everyone who participates in a Scrum team plays a critical role in helping to develop a collaborative, cross-functional approach. When people act from a common sense of values, it helps to unify everyone on the team, reduces potential conflict, and provides a solid foundation for guiding the team.

DISCUSSION TOPICS

Scrum Overview

1. Explain the difference between an empirical process control model and a defined and predictive process control model. When would each make sense?

Scrum Roles

2. How do you think the overall need for project management changes in a Scrum project, and how would the functions of a traditional project manager be absorbed into the Scrum roles on a typical Scrum team?

3. What are some of the most important differences between Scrum roles and traditional project roles, and why is the shift in roles important in an agile environment?

Scrum Methodology

4. What is the role of the product backlog in a Scrum project? How does it differ from the requirements definition practices in a traditional project?

5. When is the sprint planning meeting held? What are the major parts of a sprint planning meeting? What is the output of the sprint planning meeting?
6. What is the purpose of the sprint review meeting, and when is it held? What is the desired output of the sprint review meeting?
7. What is the objective of the sprint retrospective meeting, and how is it different from a sprint review meeting?
8. Who attends the daily standup meeting, and what is its purpose?

General Scrum/Agile Principles

9. Which general Scrum/agile principle do you think might have had the greatest impact on projects you have been involved in, and why?
10. Discuss a project and how you might have used the general Scrum/agile principles to do the project differently. Why?
11. Which general Scrum/agile principles might be the most difficult to implement, and why?

Scrum Values

12. Why are values important? How does having a consistent set of values affect the performance of a Scrum team?
13. Which Scrum value is likely to have the most impact on team performance? Why?
14. If a team is having conflict, what might that indicate? What action would be appropriate to resolve the conflict?