Chapter 2. Scrum Framework

This chapter provides an overview of the Scrum framework with a primary focus on its practices, including roles, activities, and artifacts. Subsequent chapters will provide a deeper treatment of each of these practices, including an in-depth look at the principles that underlie the practices.

Overview

Scrum is not a standardized process where you methodically follow a series of sequential steps that are guaranteed to produce, on time and on budget, a high-quality product that delights customers. Instead, Scrum is a **framework** for organizing and managing work. The Scrum framework is based on a set of values, principles, and practices that provide the foundation to which your organization will add its unique implementation of relevant engineering practices and your specific approaches for realizing the Scrum practices. The result will be a version of Scrum that is uniquely yours.

To better grasp the framework concept, imagine that the Scrum framework is like the foundation and walls of a building. The Scrum values, principles, and practices would be the key structural components. You can't ignore or fundamentally change a value, principle, or practice without risking collapse. What you can do, however, is customize inside the structure of Scrum, adding fixtures and features until you have a process that works for you.

Scrum is a refreshingly simple, people-centric framework based on the values of honesty, openness, courage, respect, focus, trust, empowerment, and collaboration. Chapter 3 will describe the Scrum principles in depth; subsequent chapters will highlight how specific practices and approaches are rooted in these principles and values.

The Scrum practices themselves are embodied in specific roles, activities, artifacts, and their associated rules (see Figure 2.1).

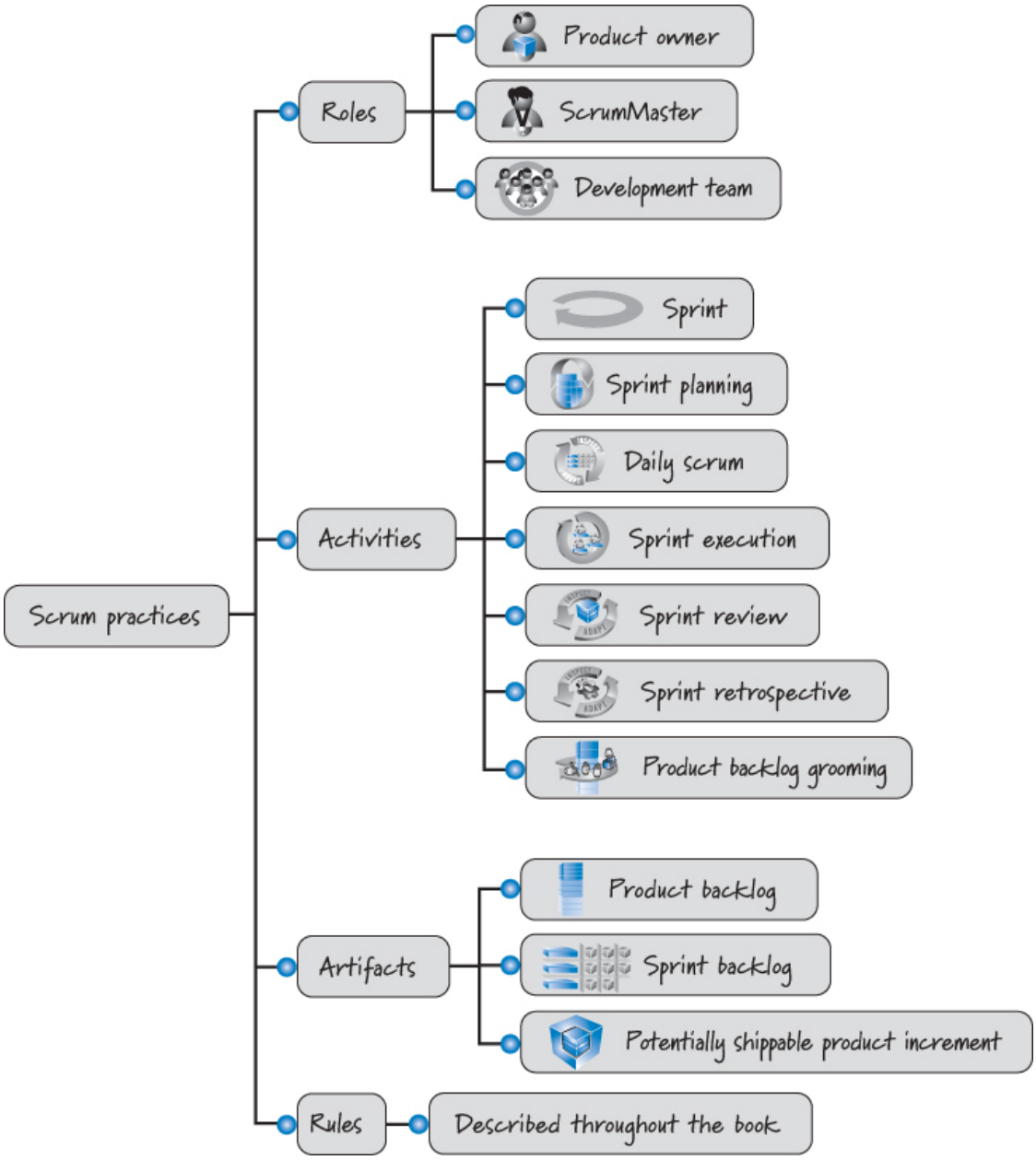

Figure 2.1. Scrum practices

The remainder of this chapter will focus on Scrum practices.

Scrum Roles

Scrum development efforts consist of one or more **Scrum teams**, each made up of three Scrum roles: **product owner**, **ScrumMaster**, and the **development team** (see Figure 2.2). There can be other roles when using Scrum, but the Scrum framework requires only the three listed here.

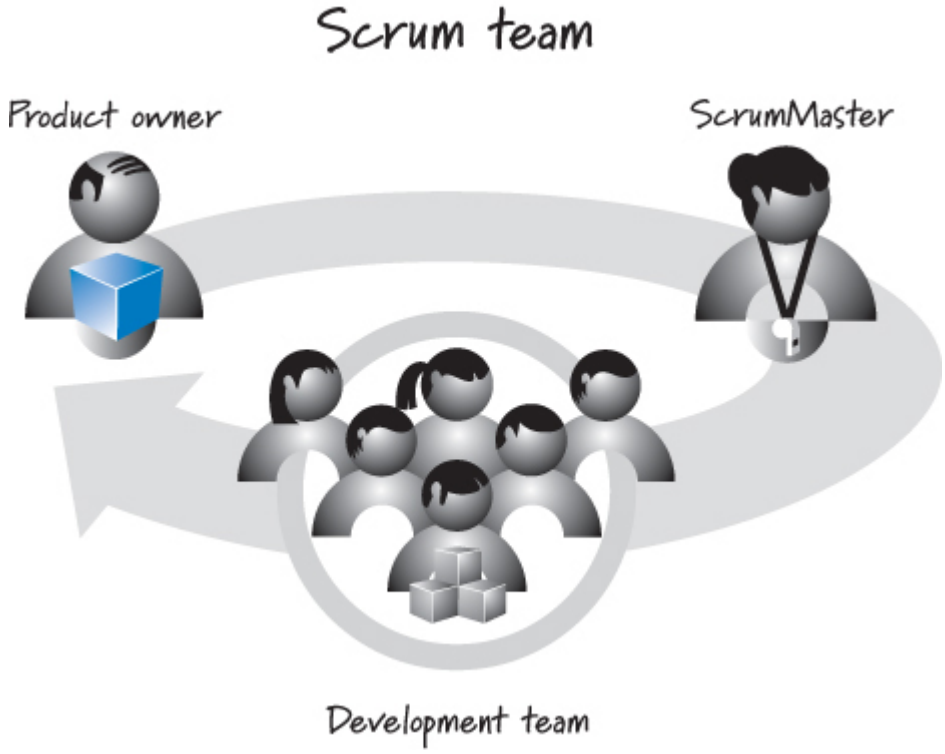

Figure 2.2. Scrum roles

The product owner is responsible for what will be developed and in what order. The ScrumMaster is responsible for guiding the team in creating and following its own process based on the broader Scrum framework. The development team is responsible for determining how to deliver what the product owner has asked for.

If you are a manager, don't be concerned that "manager" doesn't appear as a role in Figure 2.2; managers still have an important role in organizations that use Scrum (see Chapter 13). The Scrum framework defines just the roles that are specific to Scrum, not all of the roles that can and should exist within an organization that uses Scrum.

Product Owner

The product owner is the empowered central point of product leadership. He[1] is the single authority responsible for deciding which features and functionality to build and the order in which to build them. The product owner maintains and communicates to all other participants a clear vision of what the Scrum team is trying to achieve. As such, the product

owner is responsible for the overall success of the solution being developed or maintained.

It doesn't matter if the focus is on an external product or an internal application; the product owner still has the obligation to make sure that the most valuable work possible, which can include technically focused work, is always performed. To ensure that the team rapidly builds what the product owner wants, the product owner actively collaborates with the ScrumMaster and development team and must be available to answer questions soon after they are posed. See Chapter 9 for a detailed description of the product owner role.

ScrumMaster

The ScrumMaster helps everyone involved understand and embrace the Scrum values, principles, and practices. She acts as a coach, providing process leadership and helping the Scrum team and the rest of the organization develop their own high-performance, organization-specific Scrum approach. At the same time, the ScrumMaster helps the organization through the challenging change management process that can occur during a Scrum adoption.

As a facilitator, the ScrumMaster helps the team resolve issues and make improvements to its use of Scrum. She is also responsible for protecting the team from outside interference and takes a leadership role in removing **impediments** that inhibit team productivity (when the individuals themselves cannot reasonably resolve them). The ScrumMaster has no authority to exert control over the team, so this role is not the same as the traditional role of project manager or development manager. The ScrumMaster functions as a leader, not a manager. I will discuss the roles of functional manager and project manager in Chapter 13. See Chapter 10 for more details on the ScrumMaster role.

Development Team

Traditional software development approaches discuss various job types, such as architect, programmer, tester, database administrator, UI designer, and so on. Scrum defines the role of a development team, which is simply a diverse, cross-functional collection of these types of people who are responsible for designing, building, and testing the desired product.

The development team self-organizes to determine the best way to accomplish the goal set out by the product owner. The development team is typically five to nine people in size; its members must collectively have all of the skills needed to produce good-quality, working software. Of course, Scrum can be used on development efforts that require much larger teams. However, rather than having one Scrum team with, say, 35 people, there would more likely be four or more Scrum teams, each with a development team of nine or fewer people. See Chapter 11 for more details on the development team role and Chapter 12 for more details on coordinating multiple teams.

Scrum Activities and Artifacts

Figure 2.3 illustrates most of the Scrum activities and artifacts and how they fit together.

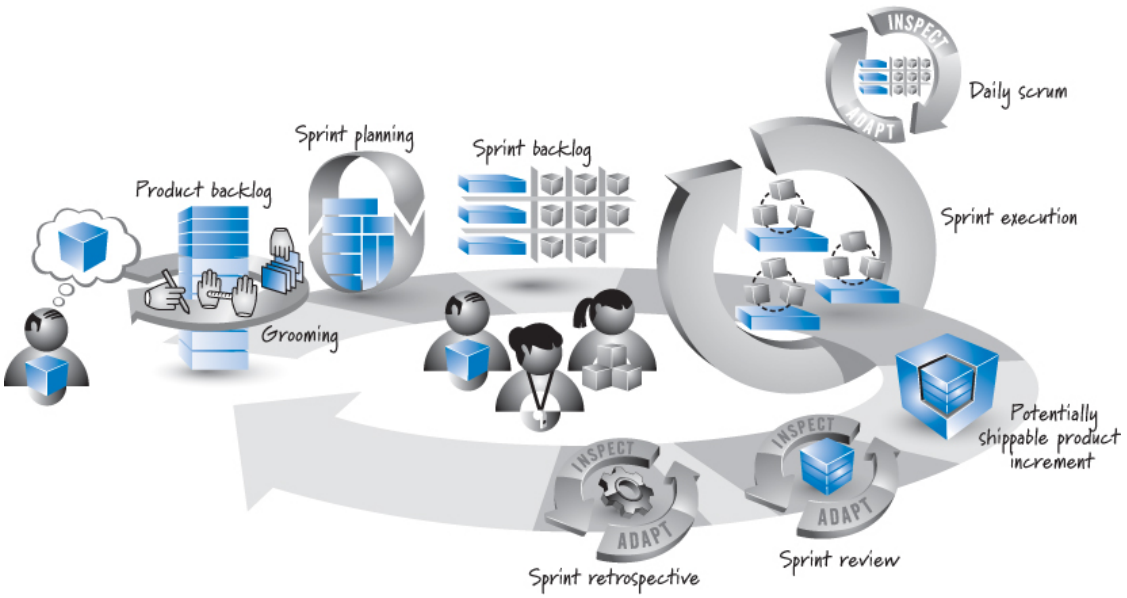

Figure 2.3. Scrum framework

Let's summarize the diagram, starting on the left side of the figure and working clockwise around the main looping arrow (the sprint).

The product owner has a vision of what he wants to create (the big cube). Because the cube can be large, through an activity called **grooming** it is broken down into a set of features that are collected into a prioritized list called the product backlog.

A sprint starts with sprint planning, encompasses the development work during the sprint (called sprint execution), and ends with the review and retrospective. The sprint is represented by the large, looping arrow that dominates the center of the figure. The number of items in the product backlog is likely to be more than a development team can complete in a short-duration sprint. For that reason, at the beginning of each sprint, the development team must determine a subset of the product backlog items it believes it can complete—an activity called sprint planning, shown just to the right of the large product backlog cube.

As a brief aside, in 2011 a change in "The Scrum Guide" (Schwaber and Sutherland 2011) generated debate about whether the appropriate term for describing the result of sprint planning is a **forecast** or a **commitment**. Advocates of the word *forecast* like it because they feel that although the development team is making the best estimate that it can at the time, the estimate might change as more information becomes known during the course of the sprint. Some also believe that a commitment on the part of the team will cause the team to sacrifice quality to meet the commitment or will cause the team to "under-commit" to guarantee that the commitment is met.

I agree that all development teams should generate a forecast (estimate) of what they can deliver each sprint. However, many development teams would benefit from using the forecast to derive a commitment. Commitments support mutual trust between the product owner and the development team as well as within the development team. Also, commitments support reasonable short-term planning and decision making

within an organization. And, when performing multiteam product development, commitments support synchronized planning—one team can make decisions based on what another team has committed to do. In this book, I favor the term *commitment*; however, I occasionally use *forecast* if it seems correct in context.

To acquire confidence that the development team has made a reasonable commitment, the team members create a second backlog during sprint planning, called the sprint backlog. The sprint backlog describes, through a set of detailed **tasks**, how the team plans to design, build, integrate, and test the selected subset of features from the product backlog during that particular sprint.

Next is sprint execution, where the development team performs the tasks necessary to realize the selected features. Each day during sprint execution, the team members help manage the flow of work by conducting a synchronization, inspection, and adaptive planning activity known as the daily scrum. At the end of sprint execution the team has produced a potentially shippable product increment that represents some, but not all, of the product owner's vision.

The Scrum team completes the sprint by performing two inspect-and-adapt activities. In the first, called the sprint review, the stakeholders and Scrum team inspect the product being built. In the second, called the sprint retrospective, the Scrum team inspects the Scrum process being used to create the product. The outcome of these activities might be adaptations that will make their way into the product backlog or be included as part of the team's development process.

At this point the Scrum sprint cycle repeats, beginning anew with the development team determining the next most important set of product backlog items it can complete. After an appropriate number of sprints have been completed, the product owner's vision will be realized and the solution can be released.

In the remainder of this chapter I will discuss each of these activities and artifacts in greater detail.

Product Backlog

Using Scrum, we always do the most valuable work first. The product owner, with input from the rest of the Scrum team and stakeholders, is ultimately responsible for determining and managing the sequence of this work and communicating it in the form of a prioritized (or ordered) list known as the **product backlog** (see Figure 2.4). On new-product development the product backlog items initially are features required to meet the product owner's vision. For ongoing product development, the product backlog might also contain new features, changes to existing features, defects needing repair, technical improvements, and so on.

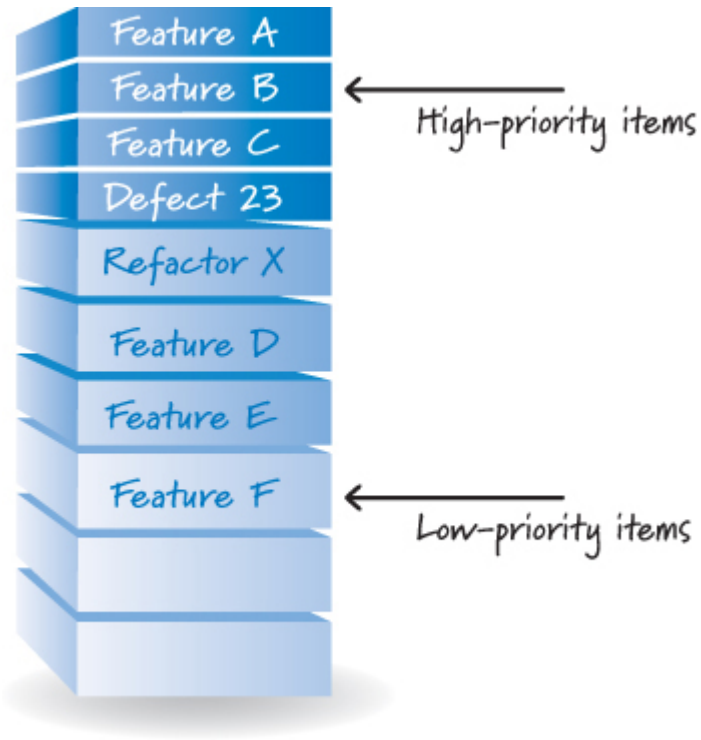

Figure 2.4. Product backlog

The product owner collaborates with internal and external stakeholders to gather and define the product backlog items. He then ensures that product backlog items are placed in the correct sequence (using factors such as value, cost, knowledge, and risk) so that the high-value items appear at the top of the product backlog and the lower-value items appear

toward the bottom. The product backlog is a constantly evolving artifact. Items can be added, deleted, and revised by the product owner as business conditions change, or as the Scrum team's understanding of the product grows (through feedback on the software produced during each sprint).

Overall the activity of creating and refining product backlog items, estimating them, and prioritizing them is known as grooming (see Figure 2.5).

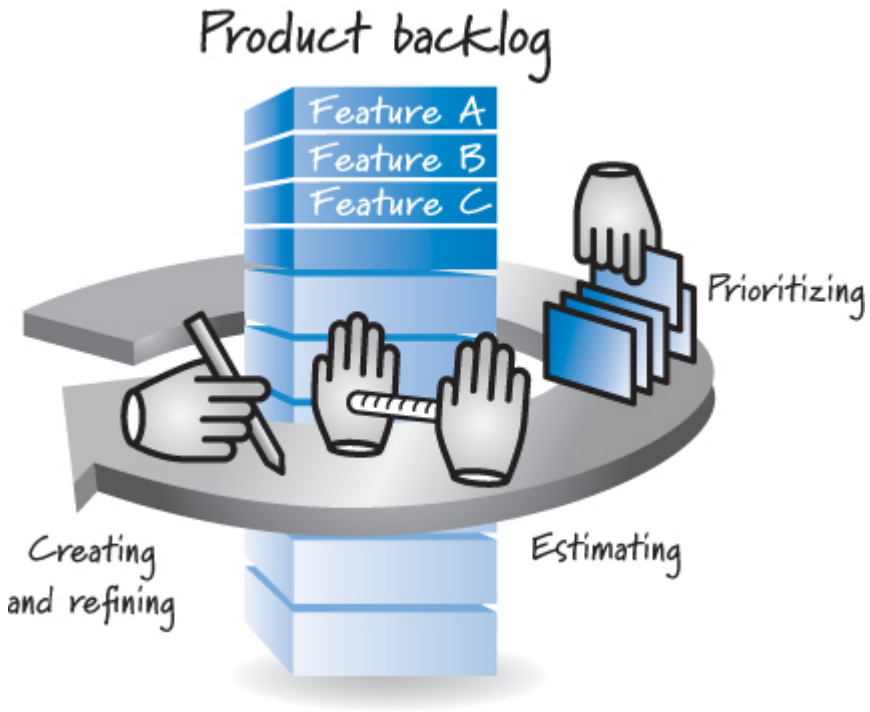

Figure 2.5. Product backlog grooming

As a second brief aside, in 2011 there was another debate as to whether the appropriate term for describing the sequence of items in the product backlog should be *prioritized* (the original term) or *ordered*, the term used in "The Scrum Guide" (Schwaber and Sutherland 2011). The argument was that prioritizing is simply one form of ordering (and, according to some, not even the most appropriate form of ordering). The issue of how to best sequence items in the product backlog, however, is influenced by many factors, and a single word may never capture the full breadth and depth of the concept. Although there may be theoretical merit to the or-

dered-versus-prioritized debate, most people (including me) use the terms interchangeably when discussing the items in the product backlog.

Before we finalize prioritizing, ordering, or otherwise arranging the product backlog, we need to know the size of each item in the product backlog (see Figure 2.6).

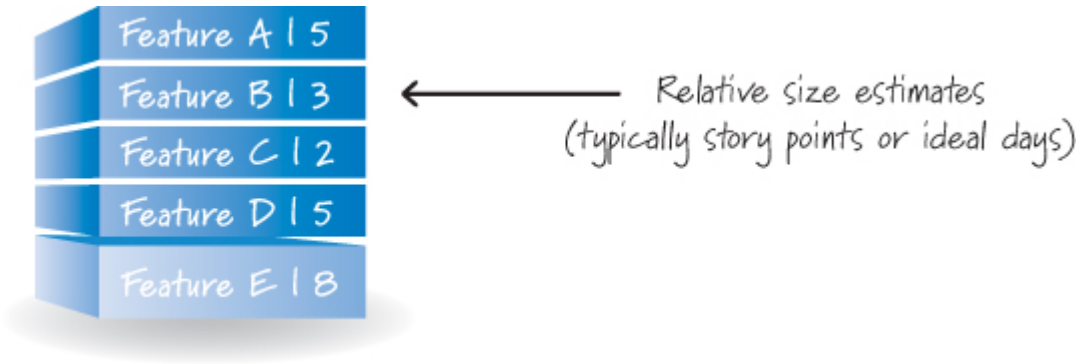

Figure 2.6. Product backlog item sizes

Size equates to cost, and product owners need to know an item's cost to properly determine its priority. Scrum does not dictate which, if any, size measure to use with product backlog items. In practice, many teams use a **relative size measure** such as **story points** or **ideal days**. A relative size measure expresses the overall size of an item in such a way that the absolute value is not considered, but the relative size of an item compared to other items is considered. For example, in Figure 2.6, feature C is size 2 and feature E is size 8. What we can conclude is that feature E is about four times larger than feature C. I will discuss these measures further in Chapter 7.

Sprints

In Scrum, work is performed in iterations or cycles of up to a calendar month called **sprints** (see Figure 2.7). The work completed in each sprint should create something of tangible value to the customer or user.

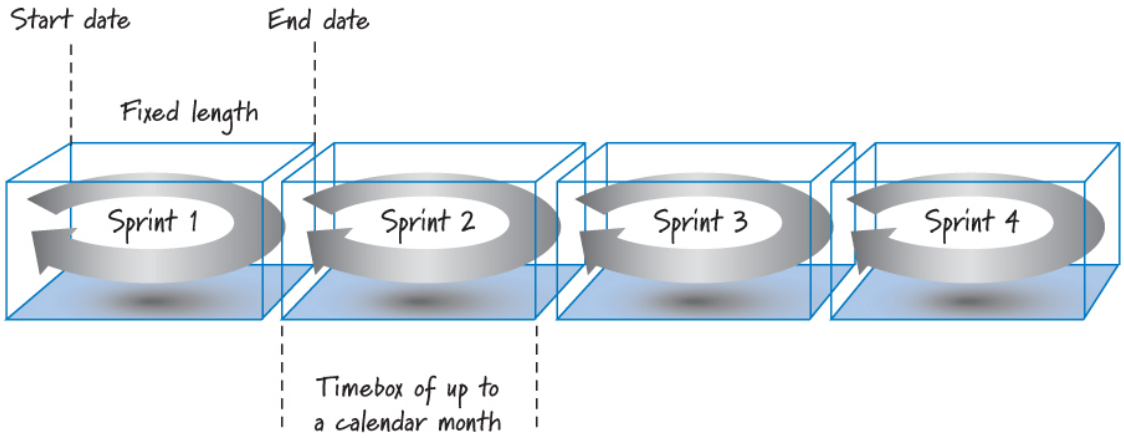

Figure 2.7. Sprint characteristics

Sprints are **timeboxed** so they always have a fixed start and end date, and generally they should all be of the same duration. A new sprint immediately follows the completion of the previous sprint. As a rule we do not permit any goal-altering changes in scope or personnel during a sprint; however, business needs sometimes make adherence to this rule impossible. I will describe sprints in more detail in Chapter 4.

Sprint Planning

A product backlog may represent many weeks or months of work, which is much more than can be completed in a single, short sprint. To determine the most important subset of product backlog items to build in the next sprint, the product owner, development team, and ScrumMaster perform **sprint planning** (see Figure 2.8).

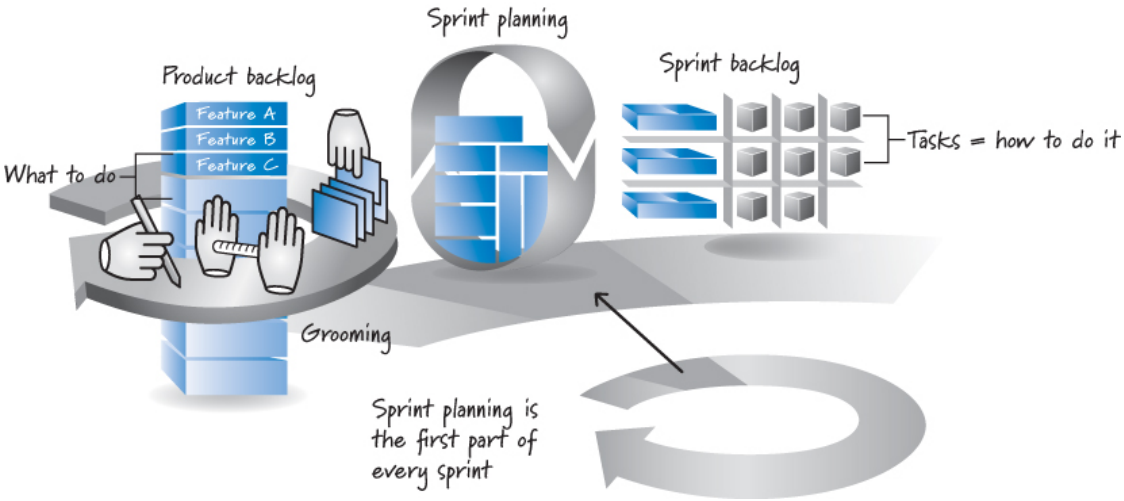

Figure 2.8. Sprint planning

During sprint planning, the product owner and development team agree on a **sprint goal** that defines what the upcoming sprint is supposed to achieve. Using this goal, the development team reviews the product backlog and determines the high-priority items that the team can realistically accomplish in the upcoming sprint while working at a **sustainable pace**—a pace at which the development team can comfortably work for an extended period of time.

To acquire confidence in what it can get done, many development teams break down each targeted feature into a set of tasks. The collection of these tasks, along with their associated product backlog items, forms a second backlog called the **sprint backlog** (see Figure 2.9).

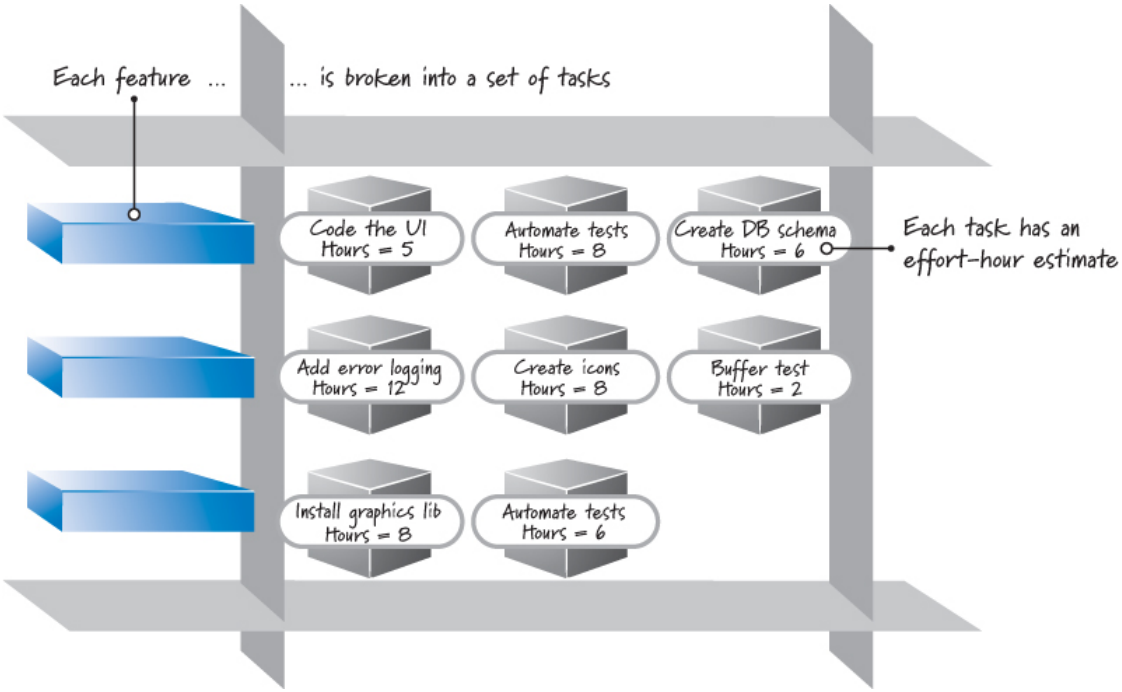

Figure 2.9. Sprint backlog

The development team then provides an estimate (typically in hours) of the effort required to complete each task. Breaking product backlog items into tasks is a form of design and **just-in-time** planning for how to get the features done.

Most Scrum teams performing sprints of two weeks to a month in duration try to complete sprint planning in about four to eight hours. A one-week sprint should take no more than a couple of hours to plan (and

probably less). During this time there are several approaches that can be used. The approach I use most often follows a simple cycle: Select a product backlog item (whenever possible, the next-most-important item as defined by the product owner), break the item down into tasks, and determine if the selected item will reasonably fit within the sprint (in combination with other items targeted for the same sprint). If it does fit and there is more capacity to complete work, repeat the cycle until the team is out of capacity to do any more work.

An alternative approach would be for the product owner and team to select all of the target product backlog items at one time. The development team alone does the task breakdowns to confirm that it really can deliver all of the selected product backlog items. I will describe each approach in more detail in Chapter 19.

Sprint Execution

Once the Scrum team finishes sprint planning and agrees on the content of the next sprint, the development team, guided by the ScrumMaster's coaching, performs all of the task-level work necessary to get the features done (see Figure 2.10), where "done" means there is a high degree of confidence that all of the work necessary for producing good-quality features has been completed.

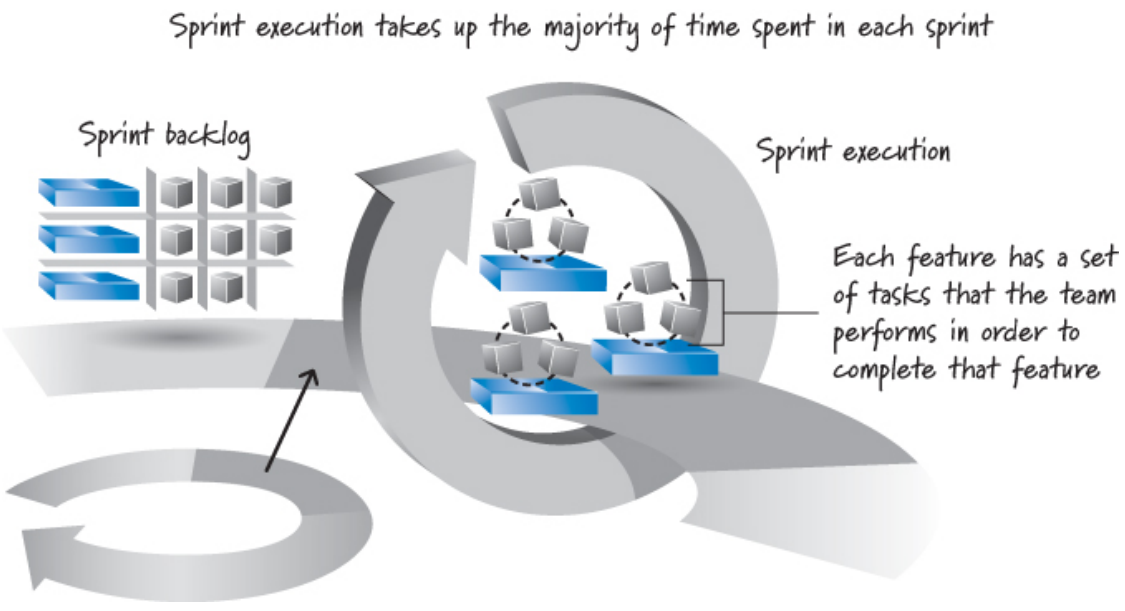

Figure 2.10. Sprint execution

Exactly what tasks the team performs depends of course on the nature of the work (for example, are we building software and what type of software, or are we building hardware, or is this marketing work?).

Nobody tells the development team in what order or how to do the task-level work in the sprint backlog. Instead, team members define their own task-level work and then self-organize in any manner they feel is best for achieving the sprint goal. See Chapter 20 for more details on sprint execution.

Daily Scrum

Each day of the sprint, ideally at the same time, the development team members hold a timeboxed (15 minutes or less) **daily scrum** (see Figure 2.11). This inspect-and-adapt activity is sometimes referred to as the **daily stand-up** because of the common practice of everyone standing up during the meeting to help promote brevity.

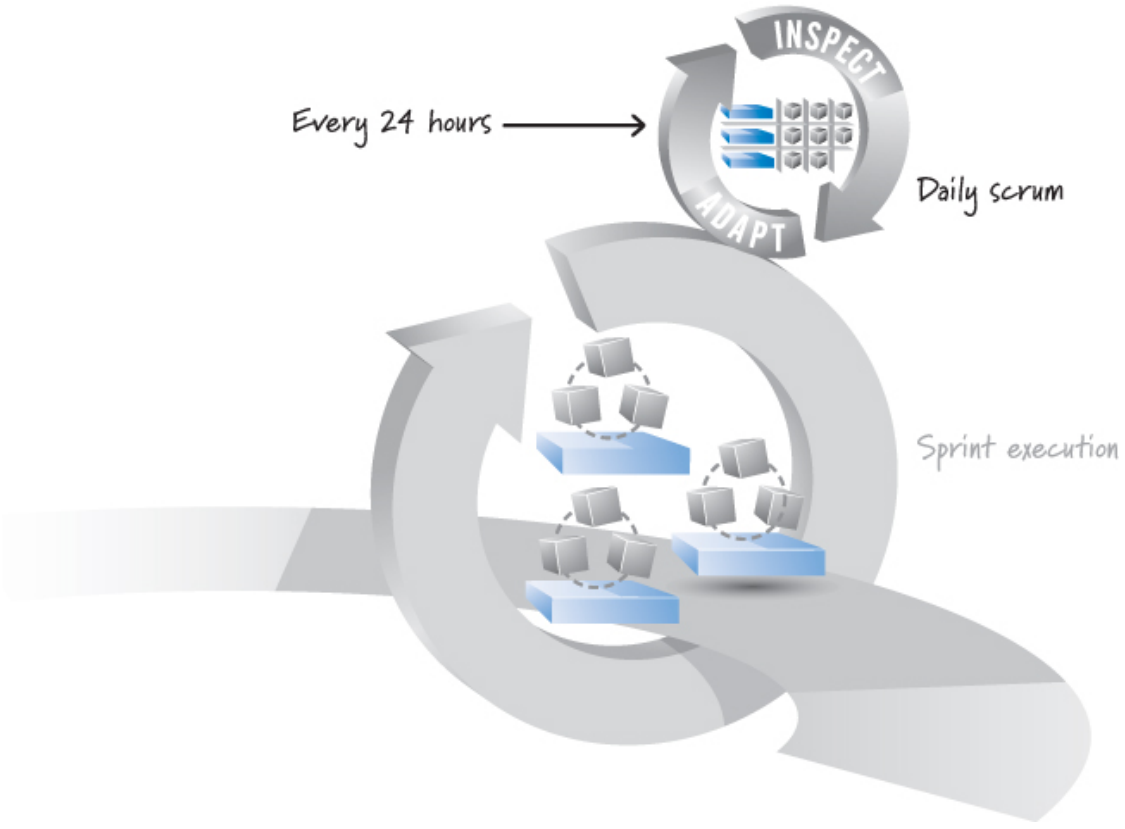

Figure 2.11. Daily scrum

A common approach to performing the daily scrum has the ScrumMaster facilitating and each team member taking turns answering three questions for the benefit of the other team members:

- What did I accomplish since the last daily scrum?
- What do I plan to work on by the next daily scrum?
- What are the obstacles or impediments that are preventing me from making progress?

By answering these questions, everyone understands the big picture of what is occurring, how they are progressing toward the sprint goal, any modifications they want to make to their plans for the upcoming day's work, and what issues need to be addressed. The daily scrum is essential for helping the development team manage the fast, flexible flow of work within a sprint.

The daily scrum is not a problem-solving activity. Rather, many teams decide to talk about problems after the daily scrum and do so with a small group of interested people. The daily scrum also is not a traditional status meeting, especially the kind historically called by project managers so that they can get an update on the project's status. A daily scrum, however, can be useful to communicate the status of sprint backlog items among the development team members. Mainly, the daily scrum is an inspection, synchronization, and adaptive daily planning activity that helps a self-organizing team do its job better.

Although their use has fallen out of favor, Scrum has used the terms "**pigs**" and "**chickens**" to distinguish who should participate during the daily scrum versus who simply observes. The farm animals were borrowed from an old joke (which has several variants): "In a ham-and-eggs breakfast, the chicken is involved, but the pig is committed." Obviously the intent of using these terms in Scrum is to distinguish between those who are involved (the chickens) and those who are committed to meeting

the sprint goal (the pigs). At the daily scrum, only the pigs should talk; the chickens, if any, should attend as observers.

I have found it most useful to consider everyone on the Scrum team a pig and anyone who isn't, a chicken. Not everyone agrees. For example, the product owner is not required to be at the daily scrum, so some consider him to be a chicken (the logic being, how can you be "committed" if you aren't required to attend?). This seems wrong to me, because I can't imagine how the product owner, as a member of the Scrum team, is any less committed to the outcome of a sprint than the development team. The metaphor of pigs and chickens breaks down if you try to apply it within a Scrum team.

Done

In Scrum, we refer to the sprint results as a **potentially shippable product increment** (see Figure 2.12), meaning that whatever the Scrum team agreed to do is really done according to its agreed-upon definition of done. This definition specifies the degree of confidence that the work completed is of good quality and is potentially shippable. For example, when developing software, a bare-minimum definition of done should yield a complete slice of product functionality that is designed, built, integrated, tested, and documented.

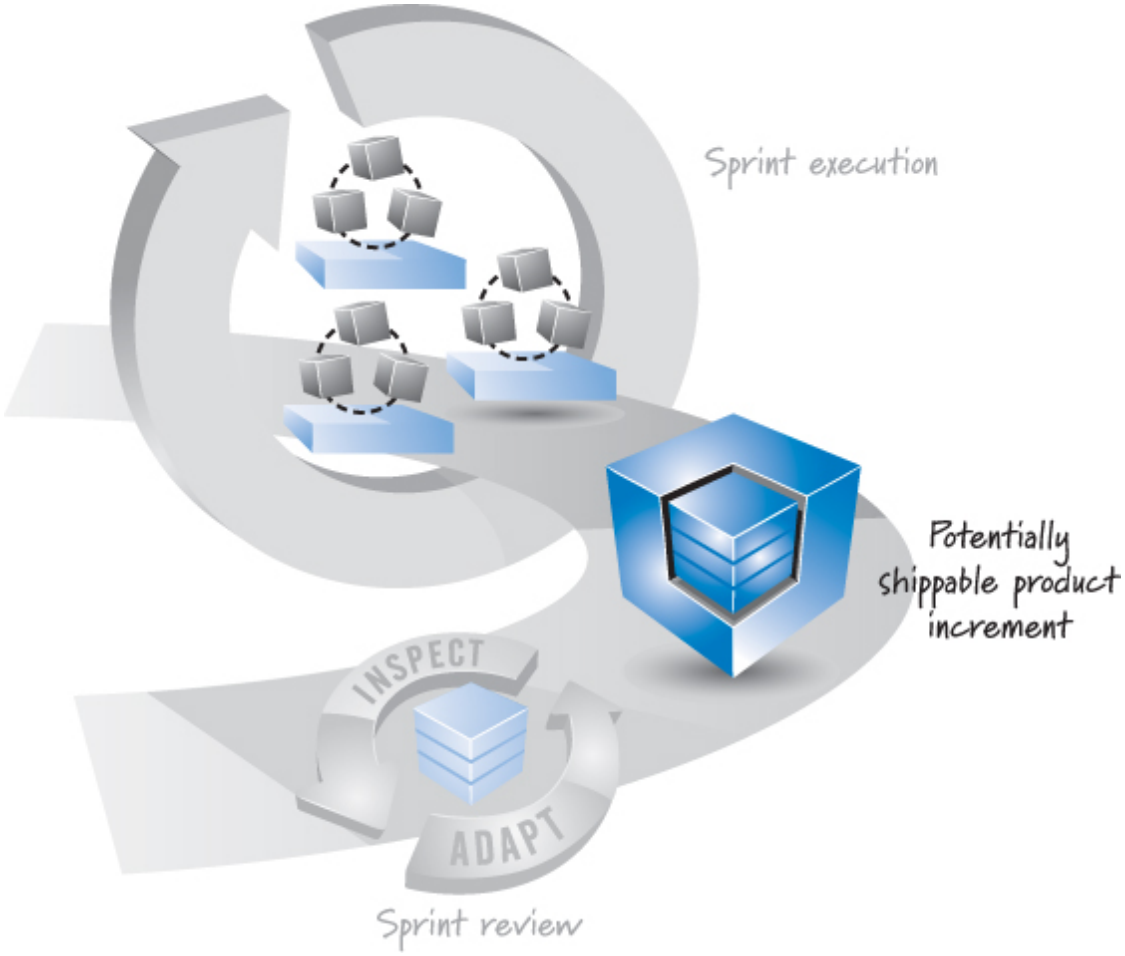

Figure 2.12. Sprint results (potentially shippable product increment)

An aggressive definition of done enables the business to decide each sprint if it wants to ship (or deploy or release) what got built to internal or external customers.

To be clear, “potentially shippable” does not mean that what got built must actually be shipped. Shipping is a business decision, which is frequently influenced by things such as “Do we have enough features or enough of a customer workflow to justify a customer deployment?” or “Can our customers absorb another change given that we just gave them a release two weeks ago?”

Potentially shippable is better thought of as a state of confidence that what got built in the sprint is actually done, meaning that there isn’t materially important undone work (such as important testing or integration and so on) that needs to be completed before we can ship the results from the sprint, if shipping is our business desire.

As a practical matter, over time some teams may vary the definition of done. For example, in the early stages of game development, having features that are potentially shippable might not be economically feasible or desirable (given the exploratory nature of early game development). In these situations, an appropriate definition of done might be a slice of product functionality that is sufficiently functional and usable to generate feedback that enables the team to decide what work should be done next or how to do it. See Chapter 4 for more details on the definition of done.

Sprint Review

At the end of the sprint there are two additional inspect-and-adapt activities. One is called the **sprint review** (see Figure 2.13).

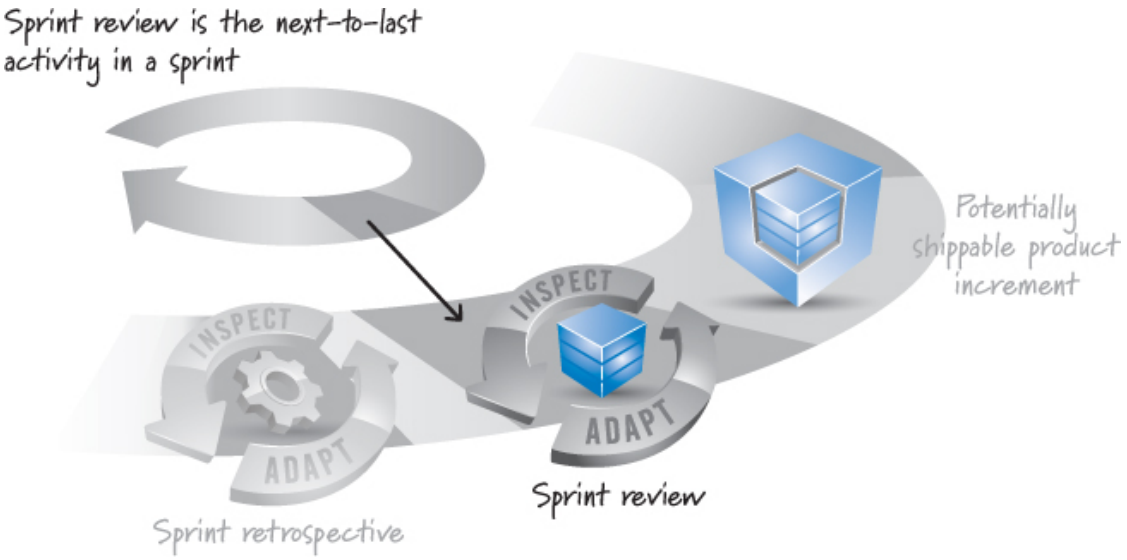

Figure 2.13. Sprint review

The goal of this activity is to inspect and adapt the product that is being built. Critical to this activity is the conversation that takes place among its participants, which include the Scrum team, stakeholders, sponsors, customers, and interested members of other teams. The conversation is focused on reviewing the just-completed features in the context of the overall development effort. Everyone in attendance gets clear visibility into what is occurring and has an opportunity to help guide the forthcoming development to ensure that the most business-appropriate solution is created.

A successful review results in bidirectional information flow. The people who aren't on the Scrum team get to sync up on the development effort and help guide its direction. At the same time, the Scrum team members gain a deeper appreciation for the business and marketing side of their product by getting frequent feedback on the convergence of the product toward delighted customers or users. The sprint review therefore represents a scheduled opportunity to inspect and adapt the product. As a matter of practice, people outside the Scrum team can perform intra-sprint feature reviews and provide feedback to help the Scrum team better achieve its sprint goal. See Chapter 21 for more details on the sprint review.

Sprint Retrospective

The second inspect-and-adapt activity at the end of the sprint is the **sprint retrospective** (see Figure 2.14). This activity frequently occurs after the sprint review and before the next sprint planning.

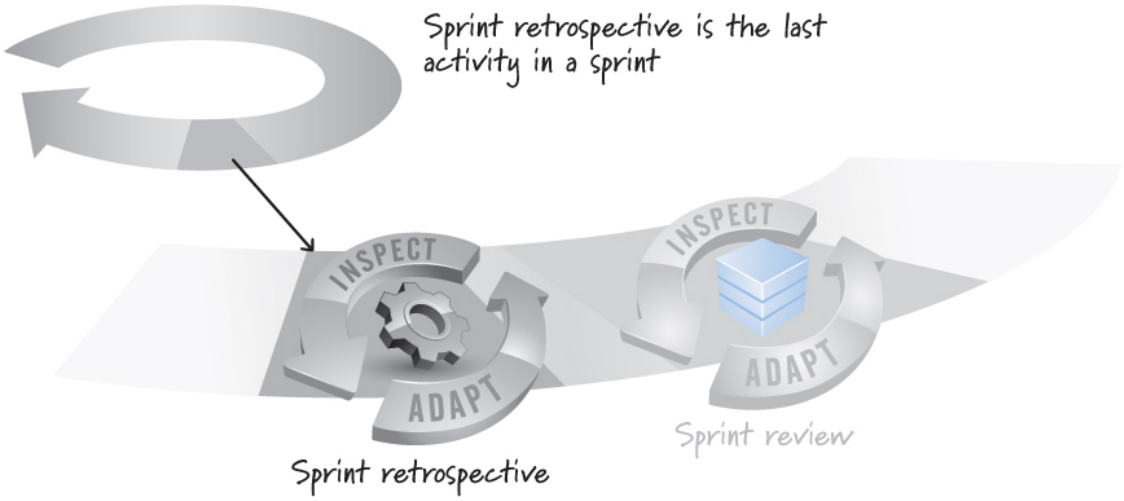

Figure 2.14. Sprint retrospective

Whereas the sprint review is a time to inspect and adapt the product, the sprint retrospective is an opportunity to inspect and adapt the process. During the sprint retrospective the development team, ScrumMaster, and product owner come together to discuss what is and is not working with Scrum and associated technical practices. The focus is on the continuous process improvement necessary to help a good Scrum team become great.

At the end of a sprint retrospective the Scrum team should have identified and committed to a practical number of process improvement actions that will be undertaken by the Scrum team in the next sprint. See Chapter 22 for details on the sprint retrospective.

After the sprint retrospective is completed, the whole cycle is repeated again—starting with the next sprint-planning session, held to determine the current highest-value set of work for the team to focus on.

Closing

This chapter described core Scrum practices, focusing on an end-to-end description of the Scrum framework's roles, activities, and artifacts. There are other practices, such as higher-level planning and progress-tracking practices, that many Scrum teams use. These will be described in subsequent chapters. In the next chapter, I will provide a description of the core principles on which Scrum is based. This will facilitate the deeper exploration of the Scrum framework in subsequent chapters.