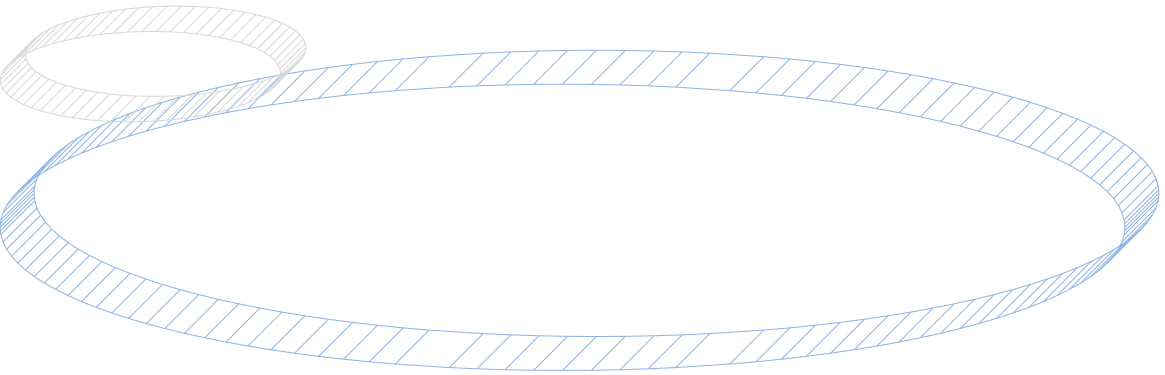

SCRUMGUIDE

November 2009

Scrum: Developed and sustained by Ken Schwaber and Jeff Sutherland

Scrum.org™

Acknowledgements

General

Scrum is based on industry-accepted best practices, used and proven for decades. It is then set in an empirical process theory. As Jim Coplien once remarked to Jeff, "Everyone will like Scrum; it is what we already do when our back is against the wall."

People

Of the thousands of people that have contributed to Scrum, we should single out those that were instrumental in its first ten years. First there were Jeff Sutherland, working with Jeff Mckenna, and Ken Schwaber with Mike Smith and Chris Martin. Scrum was first formally presented and published at OOPSLA 1995. During the next five years, Mike Beadle and Martine Devos made significant contributions. And then everyone else, without whose help Scrum wouldn't have been refined into what it is today.

History

The history of Scrum can already be considered long in the world of software development. To honor the first places where it was tried and refined, we honor Individual, Inc., Fidelity Investments, and IDX (now GE Medical).

Purpose

Scrum has been used to develop complex products since the early 1990s. This paper describes how to use Scrum to build products. Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. The role of Scrum is to surface the relative efficacy of your development practices so that you can improve upon them **while providing a framework within which complex products can be developed.**

Scrum Theory

Scrum, which is grounded in empirical process control theory, employs an iterative, incremental approach to optimize predictability and control risk. Three pillars uphold every implementation of empirical process control.

The first leg is transparency

Transparency ensures that aspects of the process that affect the outcome must be visible to those managing the outcomes. Not only must these aspects be transparent, but also what is being seen must be known. That is, when someone inspecting a process believes that something is done; it must be equivalent to their definition of done.

The second leg is inspection

The various aspects of the process must be inspected frequently enough so that unacceptable variances in the process can be detected. The frequency of inspection has to take into consideration that all processes are changed by the act of inspection. A conundrum occurs when the required frequency of inspection exceeds the tolerance to inspection of the process. Fortunately, this doesn't seem to be true of software development. The other factor is the skill and diligence of the people inspecting the work results.

The third leg is adaptation

If the inspector determines from the inspection that one or more aspects of the process are outside acceptable limits, and that the resulting product will be unacceptable, the inspector must adjust the process or the material being processed. The adjustment must be made as quickly as possible to minimize further deviation.

There are three points for inspection and adaptation in Scrum. The Daily Scrum meeting is used to inspect progress toward the Sprint goal, and to make adaptations that optimize the value of the next work day. In addition, the Sprint Review and Planning meetings are used to inspect progress toward the Release Goal and to make adaptations that optimize the value of the next Sprint. Finally, the Sprint Retrospective is used to review the past Sprint and determine what adaptations will make the next Sprint more productive, fulfilling, and enjoyable.

Scrum Content

The Scrum framework consists of a set of **Scrum Teams** and their associated roles; **Time-Boxes**, **Artifacts**, and **Rules**.

Scrum Teams are designed to optimize flexibility and productivity; to this end, they are self-organizing, they are cross-functional, and they work in iterations. Each Scrum Team has three roles: 1) the **ScrumMaster**, who is responsible for ensuring the process is understood and followed; 2) the **Product Owner**, who is responsible for maximizing the value of the work that the Scrum Team does; and 3) the **Team**, which does the work. The Team consists of developers with all the skills to turn the Product Owner's requirements into a potentially releasable piece of the product by the end of the Sprint.

Scrum employs time boxes to create regularity. Elements of Scrum that are time-boxed include the **Release Planning Meeting**, the **Sprint Planning Meeting**, the **Sprint**, the **Daily Scrum**, the **Sprint Review**, and the **Sprint Retrospective**. The heart of Scrum is a **Sprint**, which is an iteration of one month or less that is of consistent

length throughout a development effort. All Sprints use the same Scrum framework, and all Sprints deliver an increment of the final product that is potentially releasable. One Sprint starts immediately after the other.

Scrum employs four principal artifacts. The **Product Backlog** is a prioritized list of everything that might be needed in the product. The **Sprint Backlog** is a list of tasks to turn the Product Backlog for one Sprint into an increment of potentially shippable product. A burndown is a measure of remaining backlog over time. A **Release Burndown** measures remaining Product Backlog across the time of a release plan. A **Sprint Burndown** measures remaining **Sprint Backlog** items across the time of a Sprint.

Rules bind together Scrum's time-boxes, roles, and artifacts. Its rules are described throughout the body of this document. For example, it is a Scrum rule that only Team members - the people committed to turning the Product Backlog into an increment – can talk during a Daily Scrum. Ways of implementing Scrum that are not rules but rather are suggestions are described in "Tips" boxes.

Tip

When rules are not stated, the users of Scrum are expected to figure out what to do. Don't try to figure out a perfect solution, because the problem usually changes quickly. Instead, try something and see how it works. The inspect-and-adapt mechanisms of Scrum's empirical nature will guide you.

Scrum Roles

The Scrum Team consists of the ScrumMaster, the Product Owner, and the Team. Scrum Team members are called "pigs." Everyone else is a "chicken." Chickens cannot tell "pigs" how to do their work. Chickens and pigs come from the story,

"A chicken and a pig are together when the chicken says, "Let's start a restaurant!"

The pig thinks it over and says, "What would we call this restaurant?"

The chicken says, "Ham n' Eggs!"

The pig says, "No thanks, I'd be committed, but you'd only be involved!"

The ScrumMaster

The ScrumMaster is responsible for ensuring that the Scrum Team adheres to Scrum values, practices, and rules. The ScrumMaster helps the Scrum Team and the organization adopt Scrum. The ScrumMaster teaches the Scrum Team by coaching and by leading it to be more productive and produce higher quality products. The ScrumMaster helps the Scrum Team understand and use self-management and cross-functionality. The ScrumMaster also helps the Scrum Team do its best in an organizational environment that may not yet be optimized for complex product development. When the ScrumMaster helps make these changes, this is called "removing impediments. However, the ScrumMaster does not manage the Scrum Team; the Scrum Team is self-organizing.

Tip

The ScrumMaster works with the customers and management to identify and instantiate a Product Owner. The ScrumMaster teaches the Product Owner how to do his or her job. Product Owners are expected to know how to manage to optimize value using Scrum. If they don't, we hold the ScrumMaster accountable.

Tip

The ScrumMaster may be a member of the Team; for example, a developer performing Sprint tasks. However, this often leads to conflicts when the ScrumMaster has to choose between removing impediments and performing tasks. The ScrumMaster should never be the Product Owner.

The Product Owner

The Product Owner is the one and only person responsible for managing the Product Backlog and ensuring the value of the work the Team performs. This person maintains the Product Backlog and ensures that it is visible to everyone. Everyone knows what items have the highest priority, so everyone knows what will be worked on.

> **Tip**
>
> *For commercial development, the Product Owner may be the product manager. For in-house development efforts, the Product Owner could be the manager of the business function that is being automated.*

The Product Owner is one person, not a committee. Committees may exist that advise or influence this person, but people who want to change an item's priority have to convince the Product Owner. Companies that adopt Scrum may find it influences their methods for setting priorities and requirements over time.

> **Tip**
>
> *The Product Owner can be a Team member, also doing development work. This additional responsibility may cut into the Product Owner's ability to work with stakeholders. However, the Product Owner can never be the ScrumMaster.*

For the Product Owner to succeed, everyone in the organization has to respect his or her decisions. No one is allowed to tell the Team to work from a different set of priorities, and Teams aren't allowed to listen to anyone who says otherwise. The Product Owner's decisions are visible in the content and prioritization of the Product Backlog. This visibility requires the Product Owner to do his or her best, and it makes the role of Product Owner both a demanding and a rewarding one.

The Team

Teams of developers turn Product Backlog into increments of potentially shippable functionality every Sprint. Teams are also cross-functional; Team members must have all of the skills necessary to create an increment of work. Team members often have specialized skills, such as programming, quality control, business analysis, architecture, user interface design, or data base design. However, the skills that Team member share – that is, the skill of addressing a requirement and turning it into a usable product – tend to be more important than the ones that they do not. People who refuse to code because they are architects or designers are not good fits for Teams. Everyone chips in, even if that requires learning new skills or remembering old ones. There are no titles on Teams, and there are no exceptions to this rule. Teams do not contain sub-Teams dedicated to particular domains like testing or business analysis, either.

Teams are also self-organizing. No one – not even the ScrumMaster -- tells the Team how to turn Product Backlog into increments of shippable functionality. The Team figures this out on its own. Each Team member applies his or her expertise to all of the problems. The synergy that results improves the entire Team's overall efficiency and effectiveness.

The optimal size for a Team is seven people, plus or minus two. When there are fewer than five Team members, there is less interaction and as a result less productivity gain. What's more, the Team may encounter skill constraints during parts of the Sprint and be unable to deliver a releasable piece of the product. If there are more than nine members, there is simply too much coordination required. Large Teams generate too much complexity for an empirical process to manage. However, we have encountered some successful Teams that have exceeded the upper and lower bounds of this size range. The Product Owner and ScrumMaster roles are not included in this count unless they are also pigs.

Team composition may change at the end of a Sprint. Every time Team membership is changed, the productivity gained from self-organization is diminished. Care should be taken when changing Team composition.

Time-Boxes

The Time-Boxes in Scrum are the **Release Planning Meeting**, the **Sprint**, the **Sprint Planning Meeting**, the **Sprint Review**, the **Sprint Retrospective**, and the **Daily Scrum**.

Release Planning Meeting

The purpose of release planning is to establish a plan and goals that the Scrum Teams and the rest of the organizations can understand and communicate. Release planning answers the questions, "How can we turn the vision into a winning product in best possible way? How can we meet or exceed the desired customer satisfaction and Return on Investment?" The release plan establishes the goal of the release, the highest priority Product Backlog, the major risks, and the overall features and functionality that the release will contain. It also establishes a probable delivery date and cost that should hold if nothing changes. The organization can then inspect progress and make changes to this release plan on a Sprint-by-Sprint basis.

Release planning is entirely optional. If Scrum teams start work without the meeting, the absence of its artifacts will become apparent as an impediment that needs to be resolved. Work to resolve the impediment will become an item in the Product Backlog.

Products are built iteratively using Scrum, wherein each Sprint creates an increment of the product, starting with the most valuable and riskiest. More and more Sprints create additional increments of the product. Each increment is a potentially shippable slice of the entire product. When enough increments have been created for the Product to be of value, of use to its investors, the product is released.

Most organizations already have a release planning process, and in most of these processes most of the planning is done at the beginning of the release and left unchanged as time passes. In Scrum release planning, an overall goal and probable outcomes are defined. This release planning usually requires no more than 15-20% of the time an organization consumed to build a traditional release plan. However, a Scrum release performs just-in-time planning every Sprint Review and Sprint Planning meeting, as well as daily just-in-time planning at every Daily Scrum meeting. Overall, Scrum release efforts probably consume slightly more effort than tradition release planning efforts.

Release planning requires estimating and prioritizing the Product Backlog for the Release. There are many techniques for doing so that lie outside the purview of Scrum but are nonetheless useful when used with it.

The Sprint

A Sprint is an iteration. Sprints are time-boxed. During the Sprint, the ScrumMaster ensures that no changes are made that would affect the Sprint Goal. Both Team composition and quality goals remain constant throughout the Sprint. Sprints contain and consist of the Sprint Planning meeting, the development work, the Sprint Review, and the Sprint Retrospective. Sprints occur one after another, with no time in between Sprints.

> **Tip**
>
> *If the Team senses that it has overcommitted, it meets with the Product Owner to remove or reduce the scope of Product Backlog selected for the Sprint. If the Team senses that it may have extra time, it can work with the Product Owner to select additional Product Backlog.*

A project is used to accomplish something; in software development, it is used to build a product or system. Every project consists of a definition of what is to be built, a plan to build it, the work done according to the plan, and the resultant product. Every project has a horizon, which is to say the time frame for which the plan is good. If

the horizon is too long, the definition may have changed, too many variables may have entered in, the risk may be too great, etc. Scum is a framework for a project whose horizon is no more than one month long, where there is enough complexity that a longer horizon is too risky. The predictability of the project has to be controlled at least each month, and the risk that the project may go out of control or become unpredictable is contained at least each month.

> **Tip**
>
> *When a Team begins Scrum, two-week Sprints allow it to learn without wallowing in uncertainty. Sprints of this length can be synchronized with other Teams by adding two increments together.*

Sprints 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 Team, or the ScrumMaster. Under what kind of circumstances might a Sprint need to be cancelled? Management may need to cancel a Sprint if the Sprint Goal becomes obsolete. This could occur if the company changes direction or if market or technology conditions change. In general, a Sprint should be cancelled if it no longer makes sense given the circumstances. However, because of the short duration of Sprints, it rarely makes sense to do so.

When a Sprint is cancelled, any completed and "done" Product Backlog items are reviewed. They are accepted if they represent a potentially shippable increment. All other Product Backlog items are put back on the Product Backlog with their initial estimates. Any work done on them is assumed to be lost. Sprint terminations consume resources, since everyone has to regroup in another Sprint planning meeting to start another Sprint. Sprint terminations are often traumatic to the Team, and they are very uncommon.

Sprint Planning Meeting

The Sprint Planning meeting is when the iteration is planned. It is time-boxed to eight hours for a one month Sprint. For shorter Sprints,

allocate approximately 5% of the total Sprint length to this meeting and consists of two parts. The first part, a four hour time-box, is when what will be done in the Sprint is decided upon. The second part, another four-hour time box, is when the Team figures out how it is going to build this functionality into a product increment during the Sprint.

There are two parts to the Sprint Planning Meeting: the "What?" part and the "How?" part. Some Scrum Teams combine the two. In the first part, the Scrum Team addresses the question of "What?" Here, the Product Owner presents the top priority Product Backlog to the Team. They work together to figure out what functionality is to be developed during the next Sprint. The input to this meeting is the Product Backlog, the latest increment of product, the capacity of the Team, and past performance of the Team. The amount of backlog the Team selects is solely up to the Team. Only the Team can assess what it can accomplish over the upcoming Sprint.

Having selected the Product Backlog, a Sprint Goal is crafted. The Sprint Goal is an objective that will be met through the implementation of the Product Backlog. This is a statement that provides guidance to the Team on why it is building the increment. The Sprint Goal is a subset of the release goal.

The reason for having a Sprint Goal is to give the Team some wiggle room regarding the functionality. For example, the goal for the above Sprint could also be: "Automate the client account modification functionality through a secure, recoverable transaction middleware capability." As the Team works, it keeps this goal in mind. In order to satisfy the goal, it implements the functionality and technology. If the work turns out to be harder than the Team had expected, then the Team collaborates with the Product Owner and only partially implement the functionality.

In the second part of the Sprint Planning Meeting, the Team addresses the question of "How?" During the second four hours of the Sprint Planning Meeting, the Team figures out how it will turn the Product Backlog selected during Sprint Planning Meeting (What) into a done increment. The Team usually starts by designing the work. While designing, the Team identifies tasks. These tasks are the detailed pieces of work needed to convert the Product Backlog into working software. Tasks should have decomposed so they can be done in less than one day. This task list is called the Sprint Backlog. The Team self-organizes to assign and undertake the work in the Sprint Backlog, either during the Sprint Planning meeting or just-in-time during the Sprint.

The Product Owner is present during the second part of the Sprint Planning Meeting to clarify the Product Backlog and to help make trade-offs. If the Team determines that it has too much or too little work, it may renegotiate the Product Backlog with the Product Owner. The Team may also invite other people to attend in order to provide technical or domain advice. A new Team often first realizes that it will either sink or swim as a Team, not individually, in this meeting. The Team realizes that it must rely on itself. As it realizes this, it starts to self-organize to take on the characteristics and behavior of a real Team.

> **Tip**
>
> *Usually, only 60-70% of the total Sprint Backlog will be devised in the Sprint Planning meeting. The rest is stubbed out for later detailing, or given large estimates that will be decomposed later in the Sprint.*

Sprint Review

At the end of the Sprint, a Sprint Review meeting is held. This is a four hour time-boxed meeting for one month Sprints. For Sprints of lesser duration, this meeting must not consume more than 5% of the total Sprint. During the Sprint Review, the Scrum Team and stakeholders collaborate about what was just done. Based on that and changes to

the Product Backlog during the Sprint, they collaborate about what are the next things that could be done. This is an informal meeting, with the presentation of the functionality intended to foster collaboration about what to do next.

The meeting includes at least the following elements. The Product Owner identifies what has been done and what hasn't been done. The Team discusses what went well during the Sprint and what problems it ran into, and how it solved these problems. The Team then demonstrates the work that is done and answers questions. The Product Owner then discusses the Product Backlog as it stands. He or she projects likely completion dates with various velocity assumptions. The entire group then collaborates about what it has seen and what this means regarding what to do next. The Sprint Review provides valuable input to subsequent Sprint Planning meeting.

Sprint Retrospective

After the Sprint Review and prior to the next Sprint Planning meeting, the Scrum Team has a Sprint Retrospective meeting. At this three hour, time-boxed meeting the ScrumMaster encourages the Scrum Team to revise, within the Scrum process framework and practices, their development process to make it more effective and enjoyable for the next Sprint. Many books document techniques that are helpful to use in Retrospectives.

The purpose of the Retrospective is to inspect how the last Sprint went in regards to people, relationships, process and tools. The inspection should identify and prioritize the major items that went well and those items that-if done differently-could make things even better. These include Scrum Team composition, meeting arrangements, tools, definition of "done," methods of communication, and processes for turning Product Backlog items into something "done." By the end of the Sprint Retrospective, the Scrum Team should have identified actionable improvement measures that it implements in the next Sprint. These changes become the adaptation to the empirical inspection.

Daily Scrum

Each Team meets daily for a 15-minute inspect and adapt meeting called the Daily Scrum. The Daily Scrum is at the same time and same place throughout the Sprints. During the meeting, each Team member explains:

1. What he or she has accomplished since the last meeting;
2. What he or she is going to do before the next meeting; and
3. What obstacles are in his or her way.

Daily Scrums improve communications, eliminate other meetings, identify and remove impediments to development, highlight and promote quick decision-making, and improve everyone's level of project knowledge.

The ScrumMaster ensures that the Team has the meeting. The Team is responsible for conducting the Daily Scrum. The ScrumMaster teaches the Team to keep the Daily Scrum short by enforcing the rules and making sure that people speak briefly. The ScrumMaster also enforces the rule that chickens are not allowed to talk or in anyway interfere with the Daily Scrum.

The Daily Scrum is not a status meeting. It is not for anyone but the people transforming the Product Backlog items into an increment (the Team). The Team has committed to a Sprint Goal, and to these Product Backlog items. The Daily Scrum is an inspection of the progress toward that Sprint Goal (the three questions). Follow-on meetings usually occur to make adaptations to the upcoming work in the Sprint. The intent is to optimize the probability that the Team will meet its Goal. This is a key inspect and adapt meeting in the Scrum empirical process.

Scrum Artifacts

Scrum Artifacts include the Product Backlog, the Release Burndown, the Sprint Backlog, and the Sprint Burndown.

Product Backlog and Release Burndown

The requirements for the product that the Team(s) is developing are listed in the Product Backlog. The Product Owner is responsible for the Product Backlog, its contents, its availability, and its prioritization. Product Backlog is never complete. The initial cut at developing it only 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 Backlog is dynamic in that it constantly changes to identify what the product needs to be appropriate, competitive, and useful. As long as a product exists, Product Backlog also exists.

The Product Backlog represents everything necessary to develop and launch a successful product. It is a list of all features, functions, technologies, enhancements, and bug fixes that constitute the changes that will be made to the product for future releases. Product Backlog items have the attributes of a description, priority, and estimate. Priority is driven by risk, value, and necessity. There are many techniques for assessing these attributes.

> **Tip**
>
> *Product Backlog items are usually stated as User Stories. Use Cases are appropriate as well, but they are better for use in developing life- or mission-critical software.*

Product Backlog is sorted in order of priority. Top priority Product Backlog drives immediate development activities. The higher the priority, the more urgent it is, the more it has been thought about, and the more consensus there is regarding its value. Higher priority backlog is clearer and has more detailed information than lower priority backlog. Better estimates are made based on the greater clarity and increased detail. The lower the priority, the less the detail, until you can barely make out the item.

As a product is used, as its value increases, and as the marketplace provides feedback, the product's backlog emerges into a larger and more exhaustive list. Requirements never stop changing. Product

Backlog is a living document. Changes in business requirements, market conditions, technology, and staffing cause changes in the Product Backlog. To minimize rework, only the highest priority items need to be detailed out. The Product Backlog items that will occupy the Teams for the upcoming several Sprints are fine-grained, having been decomposed so that any one item can be done within the duration of the Sprint.

Multiple Scrum Teams often work together on the same product. One Product Backlog is used to describe the upcoming work on the Product. A Product Backlog attribute that groups items is then employed. Grouping can occur by feature set, technology, or architecture, and it is often used as a way to organize work by Scrum Team.

Tip

Scrum Teams often spend 10% of each Sprint grooming the product backlog to meet the above definition of the Product Backlog. When groomed to this level of granularity, the Product Backlog items at the top of the Product Backlog (highest priority, greatest value) are decomposed so they fit within one Sprint. They have been analyzed and thought through during the grooming process. When the Sprint Planning meeting occurs, these top priority Product Backlog items are well understood and easily selected.

Tip

Acceptance tests are often used as another Product Backlog item attribute. They can often supplant more detailed text descriptions with a testable description of what the Product Backlog item must do when completed.

The Release Burndown graph records the sum of remaining Product Backlog estimated effort across time. The estimated effort is in whatever unit of work the Scrum Team and organization have decided upon. The units of time are usually Sprints.

Product Backlog item estimates are calculated initially during Release Planning, and thereafter as they are created. During Product Backlog grooming they are reviewed and revised. However, they can be updated at any time. The Team is responsible for all estimates. The

Product Owner may influence the Team by helping understand and select trade-offs, but the final estimate is made by the Team. The Product Owner keeps an updated Product Backlog list Release Backlog Burndown posted at all times. A trend line can be drawn based on the change in remaining work.

Tip

In some organizations, more work is added to the backlog than is completed. This may create a trend line that is flat or even slopes upwards. To compensate for this and retain transparency, a new floor may be created when work is added or subtracted. The floor should add or remove only significant changes and should be well documented.

Sprint Backlog and Sprint Burndown

The Sprint Backlog consists of the tasks the Team performs to turn Product Backlog items into a "done" increment. Many are developed during the Sprint Planning Meeting. It is all of the work that the Team identifies as necessary to meet the Sprint goal. Sprint Backlog items must be decomposed. The decomposition is enough so changes in progress can be understood in the Daily Scrum.

Tip

The trend line may be unreliable for the first two to three Sprints of a release unless the Teams have worked together before, know the product well, and understand the underlying technology.

The Team modifies Sprint Backlog throughout the Sprint, as well as Sprint Backlog emerging during the Sprint. As it gets into individual tasks, it may find out that more or fewer tasks are needed, or that a given task will take more or less time than had been expected. As new work is required, the Team adds it to the Sprint Backlog. As tasks are worked on or completed, the hours of estimated remaining work for each task is updated. When tasks are deemed unnecessary, they are removed. Only the Team can change its Sprint Backlog during a Sprint. Only the Team can change the contents or the estimates. The Sprint Backlog is a highly visible, real time picture of the work that the

Team plans to accomplish during the Sprint, and it belongs solely to the Team.

Sprint Backlog Burndown is a graph of the amount of Sprint Backlog work remaining in a Sprint across time in the Sprint. To create this graph, determine how much work remains by summing the backlog estimates every day of the Sprint. The amount of work remaining for a Sprint is the sum of the work remaining for all of Sprint Backlog. Keep track of these sums by day and use them to create a graph that shows the work remaining over time. By drawing a line through the points on the graph, the Team can manage its progress in completing a Sprint's work. Duration is not considered in Scrum. Work remaining and date are the only variables of interest.

> **Tip**
>
> *Whenever possible, hand draw the burndown chart on a big sheet of paper displayed in the Team's work area. Teams are more likely to see a big, visible chart than they are to look at Sprint burndown chart in Excel or a tool.*

One of Scrum's rules pertains to the purpose of each Sprint, which is to deliver increments of potentially shippable functionality that adheres to a working definition of "done."

Done

Scrum requires Teams to build an increment of product functionality every Sprint. This increment must be potentially shippable, for Product Owner may choose to immediately implement the functionality. To do so, the increment must be a complete slice of the product. It must be "done." Each increment should be additive to all prior

> **Tip**
>
> *"Undone" work is often accumulated in a Product Backlog item called "Undone Work" or "Implementation Work." As this work accumulates, the Product Backlog burndown remains more accurate than if it weren't accumulated.*

increments and thoroughly tested, ensuring that all increments work together.

In product development, asserting that functionality is done might lead someone to assume that it is at least cleanly coded, refactored, unit tested, built, and acceptance tested. Someone else might assume only that the code has been built. If everyone doesn't know what the definition of "done" is, the other two legs of empirical process control don't work. When someone describes something as *done*, everyone must understand what *done* means.

Done defines what the Team means when it commits to "doing" a Product Backlog item in a Sprint. Some products do not contain documentation, so the definition of "done" does not include documentation. A completely "done" increment includes all of the analysis, design, refactoring, programming, documentation and testing for the increment and all Product Backlog items in the increment. Testing includes unit, system, user, and regression testing, as well as non-functional tests such as performance, stability, security, and integration. Done includes any internationalization. Some Teams aren't yet able to include everything required for implementation in their definition of done. This must be clear to the Product Owner. This remaining work will have to be done before the product can be implemented and used.

FINAL THOUGHTS

Some organizations are incapable of building a complete increment within one Sprint. They may not yet have the automated testing infrastructure to complete all of the testing. In this case, two categories are created for each increment: the "done" work and the "undone" work. The "undone" work is the portion of each increment that will have to be completed at a later time. The Product Owner knows exactly what he or she is inspecting at the end of the Sprint because the increment meets the definition of "done" and the Product Owner understands the definition. "Undone" work is added to a Product Backlog item named "undone work" so it accumulates and

correctly reflects on the Release Burndown graph. This technique creates transparency in progress toward a release. The inspect and adapt in the Sprint Review is as accurate as this transparency.

For instance, if a Team is not able to do performance, regression, stability, security, and integration testing for each Product Backlog item, the proportion of this work to the work that can be done (analysis, design, refactoring, programming, documentation, unit and user testing) is calculated. Let's say that this proportion is six pieces of "done" and four pieces on "undone." If the Team finishes a Product Backlog item of six units of work (the Team is estimating based on what it knows how to "do"), four is added to the "undone work" Product Backlog item when they are finished.

Sprint by Sprint, the "undone" work of each increment is accumulated and must be addressed prior to releasing the product. This work is accumulated linearly although it actually has some sort of exponential accumulation that is dependent on each organization's characteristics. Release Sprints are added to the end of any release to complete this "undone" work. The number of Sprints is unpredictable to the degree that the accumulation of "undone" work is not linear.