SCRUM PRIMER

By Pete Deemer and Gabrielle Benefield

Pete Deemer is Chief Product Officer, Yahoo! India Research and Development. Gabrielle Benefield is Senior Director of Agile Development at Yahoo! Inc. They lead Yahoo!'s large-scale global adoption of Scrum.

A note to readers: There are many concise descriptions of Scrum available online, and this primer aims to provide the next level of detail on the practices. It is not intended as the final step in a Scrum education; teams that are considering adopting Scrum are advised to equip themselves with Ken Schwaber's Agile Project Management with Scrum *or* Agile Software Development with Scrum*, and take advantage of the many excellent Scrum training and coaching options that are available; full details are at scrumalliance.org. Our thanks go to Ken Schwaber, Dr. Jeff Sutherland, and Mike Cohn for their generous input.*

Traditional Software Development

The traditional way to build software, used by companies big and small, is commonly known as "The Waterfall". There are many variants, but it typically begins with a detailed planning phase, where the end product is carefully thought through, designed, and documented in great detail. The tasks necessary to execute the design are determined, and the work is planned using tools like Gantt charts and programs like Microsoft Project. The team arrives at an estimate of how long the project will take by adding up detailed estimates of the individual steps involved. Once stakeholders have thoroughly reviewed the plan and provided their approvals, the team starts to build. Team members complete their specialized portion of the work, and then hand it off to others in production-line fashion. Once the work is complete, it is delivered to a Quality Assurance organization, which completes testing prior to the product reaching the customer. Throughout the process, strict controls are placed on deviations from the plan, to ensure that what is produced is actually what was designed.

This approach has strengths and weaknesses. Its great strength is that it is supremely logical: think before you build, write it all down, follow a plan, and keep everything as organized as possible. It has just one great weakness: humans are involved, and humans don't work very well this way.

For example: this approach requires that the good ideas all come at the beginning of the development cycle, where they can be incorporated into the plan. But as we all know, good ideas appear spontaneously throughout the process – in the beginning, the middle, and sometimes even the day before launch, and a process that doesn't permit change will stifle this innovation. With the Waterfall approach, a great idea late in the development cycle is not a gift, it's a threat.

The Waterfall approach also places a great emphasis on writing things down as a primary method for communicating critical information. The very reasonable assumption is that if I can write down on paper as much as possible of what's in my head, it will more reliably make it into the head of everyone else on the team; plus, if it's on paper, there is tangible proof that I've done my job. The reality, though, is that most of the time, these highly detailed 100-page requirements documents just don't get read. And that's probably just as well, because when they do get read, the misunderstandings are often compounded. A written document is an incomplete abstraction of a picture I have in my head; when you read that document, you create yet another abstraction, which is now two steps away from what I'm really thinking of. It should come as no surprise that serious misunderstandings would occur.

Something else that happens when you have humans involved is the "hands-on aha" moment – the first time that you actually use the working product, and you immediately think of 20 ways you could have made it better. Unfortunately, these very valuable insights often come at the end of the development cycle, when changes are most difficult and disruptive – in other words, when doing the right thing is most expensive.

Humans also have a poor ability to predict the future. For example, the competition makes an announcement that wasn't expected. Unanticipated technical problems crop up that force a change in direction. Furthermore, people tend to be particularly bad at planning things far into the future – guessing today how you'll be spending your week eight months from now is something of a fallacy, and it's been the downfall of many a Gantt chart.

In addition, the Waterfall also tends to foster an adversarial relationship between the team-members that are handing work off from one to the next. "He's asking me to build something that's not in the spec." "She's changing her mind about what she wants." "I can't be held responsible for something I don't control." And this gets us to another observation about the

Waterfall – it's not that much fun to work within. In fact, we'd go a step further and say that the Waterfall is a cause of great misery for the people who build products, and the resulting products fall well short of expressing the creativity, skill, and passion of their creators. People aren't robots, and a process that expects them to act like robots often results in unhappy people.

A rigid, change-resistant process will also tend to produce mediocre products. Customers may get what they first ask for, but is it what they really want once they see the product begin to emerge? By gathering all the requirements up front and having them set in stone with little chance of change, the product is condemned to be only as good as the initial idea, instead of being the best it could be once the team knows more about the possibilities.

Many users of the Waterfall experience these shortcomings again and again, but it seems like such a logical approach, the natural reaction is to turn the blame inward: "If only we did it better, it would work" – if we just planned more, documented more, resisted change more, everything would work smoothly. Unfortunately, many teams find just the opposite: the harder they try, the worse it gets!

Agile Development and Scrum

The Agile family of development methodologies was born out of a belief that an approach more grounded in human reality would yield better results. Agile emphasizes building working software that people can get hands on with quickly, versus spending a lot of time writing specifications up front. Agile focuses on small, cross-functional teams empowered to make decisions, versus big hierarchies and compartmentalization by function, and Agile focuses on rapid iteration, with as much customer input along the way as possible. Often when folks learn about Agile, there's a glimmer of recognition – it sounds a lot like back in the start-up days, when we "just did it".

One of the fastest-growing Agile methods is Scrum. It was formalized over a decade ago by Ken Schwaber and Dr. Jeff Sutherland, and it's now being used by companies large and small, including Yahoo!, Microsoft, Google, Lockheed Martin, Motorola, SAP, Cisco, GE Medical, CapitalOne and the US Federal Reserve. Many teams using Scrum report significant improvements, and in some cases complete transformations, in both productivity and morale. For product developers – many of whom have been burned by the "management fad of the month club" – this is significant. Scrum is simple, powerful, and rooted in common sense.

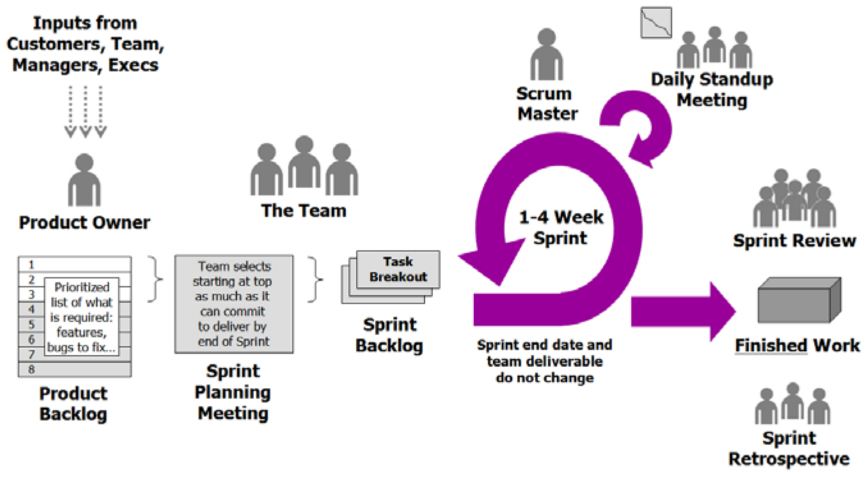

Inputs from Customers, Team, Managers, Execs
Product Owner
The Team
Scrum Master
Daily Standup Meeting
1-4 Week Sprint
Sprint Review
Prioritized list of what is required: features, bugs to fix...
Team selects starting at top as much as it can commit to deliver by end of Sprint
Task Breakout
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Sprint end date and team deliverable do not change
Finished Work
Sprint Retrospective

Figure 1. Scrum

Scrum Basics

Scrum is an iterative, incremental process. Scrum structures product development in cycles of work called **Sprints**, iterations of work which are typically 1-4 weeks in length. The Sprints are of fixed duration – they end on a specific date whether the work has been completed or not, and are never extended. At the beginning of each Sprint, a cross-functional team selects items from a prioritized list of requirements, and commits to complete them by the end of the Sprint. Each work day, the team gathers briefly to report to each other on progress, and update simple visual representations of work remaining. At the end of the Sprint, the team demonstrates what they have built, and gets feedback which can then be acted upon in the next Sprint. Scrum emphasizes producing working product that at the end of the Sprint is really "done"; in the case of software, this means code that is fully tested and potentially shippable.

Scrum Roles

In Scrum, there are three primary roles: The Product Owner, Team Members, and The ScrumMaster.

The **Product Owner** is responsible for taking all the inputs into what the product should be – from the customer or end-user of the product, as well as from Team Members and stakeholders – and translating them into a product vision. In some cases, the Product Owner and the customer are one and the same; in other cases, the customer might actually be millions of different people with a variety of needs. The Product Owner role maps to the Product Manager or Product Marketing Manager position in many organizations.

Team Members build the product that the customer is going to consume: the software, the website, or whatever it may be. The team in Scrum is typically five to ten people, although teams as large as 15 and as small as 3 commonly report benefits. The team should include all the expertise necessary to deliver the finished work – so, for example, the team for a software project might include programmers, interface designers, testers, marketers, and researchers. Team members build the product, but they also provide input and ideas to the Product Owner about how to make the product as good as it can be. Projects with more than 15 people are organized as multiple Scrum teams, each focused on a different aspect of the product development, with close coordination of their efforts. While team members can split their time with other projects, it's much more productive to have team members fully dedicated to the Scrum. Team members can also change from one Sprint to the next, but that also reduces the productivity of the team.

The **ScrumMaster** is tasked with doing whatever is necessary to help the team be successful. The ScrumMaster is not the manager of the team; he or she serves the team, by helping remove blocks to the team's success, facilitating meetings, and supporting the practice of Scrum. Some teams will have someone dedicated fully to the role of ScrumMaster, while others will have a team member play this role (carrying a lighter load of regular work when they do so). Great ScrumMasters have come from all backgrounds and disciplines: Project Management, Engineering, Design, Testing. The ScrumMaster and the Product Owner probably shouldn't be the same individual; at times, the ScrumMaster may be called upon to push back on the Product Owner (for example, if they try to introduce new requirements in the middle of a Sprint). And unlike a Project Manager, the ScrumMaster doesn't tell people what to do or assign tasks – they facilitate the process, to enable the team to organize and manage itself.

In addition to these three roles, there are other important contributors to the success of the project: Perhaps the most important of these are **Managers**. While their role evolves in Scrum, they remain critically important – they support the team in its use of Scrum, and they contribute their wisdom, expertise and assistance to the project. In Scrum, these individuals replace the time they previously spent "playing nanny" (assigning tasks, getting status reports, and other forms of micromanagement) with more time "playing guru" (mentoring, coaching, playing devil's advocate, helping remove obstacles, helping problem-solve, providing creative input, and guiding the skills development of team members). In making this shift, managers may need to evolve their management style; for example, using Socratic questioning to help the team discover the solution to a problem, rather than simply deciding a solution and assigning it to the team.

Starting Scrum

The first step in Scrum is for the Product Owner to articulate the product vision. This takes the form of a prioritized list of what's required, ranked in order of value to the customer and business, with the highest value items at the top of the list. This is called the **Product Backlog**, and it exists (and evolves) over the lifetime of the product (figure 2). The Product Backlog will include a variety of items, such as features ("enable all users to place book in shopping cart"), development requirements ("rework the transaction processing module to make it scalable"), exploratory work ("investigate solutions for speeding up credit card validation"), and known bugs ("diagnose and fix the order processing script errors").

The Product Backlog is regularly updated by the Product Owner to reflect changes in the needs of the customer, announcements by the competition, new ideas or insights, technical hurdles that appear, and so forth. At any

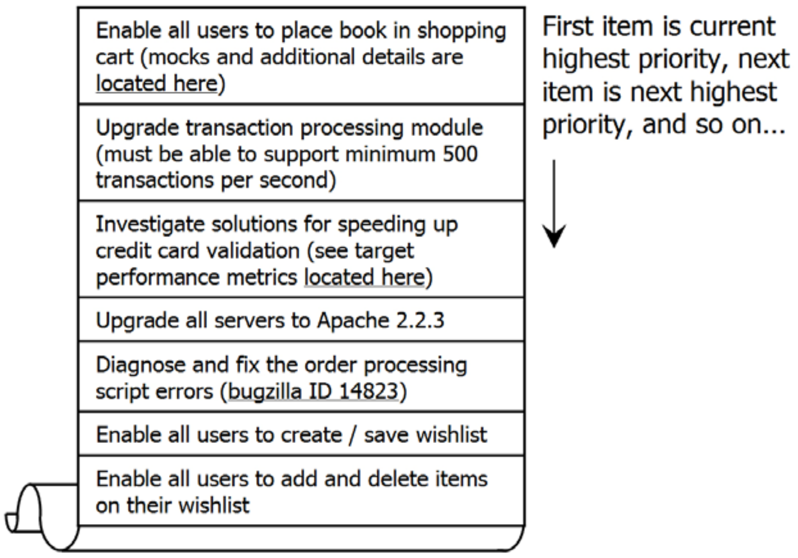

Enable all users to place book in shopping cart (mocks and additional details are located here)
Upgrade transaction processing module (must be able to support minimum 500 transactions per second)
Investigate solutions for speeding up credit card validation (see target performance metrics located here)
Upgrade all servers to Apache 2.2.3
Diagnose and fix the order processing script errors (bugzilla ID 14823)
Enable all users to create / save wishlist
Enable all users to add and delete items on their wishlist
First item is current highest priority, next item is next highest priority, and so on...

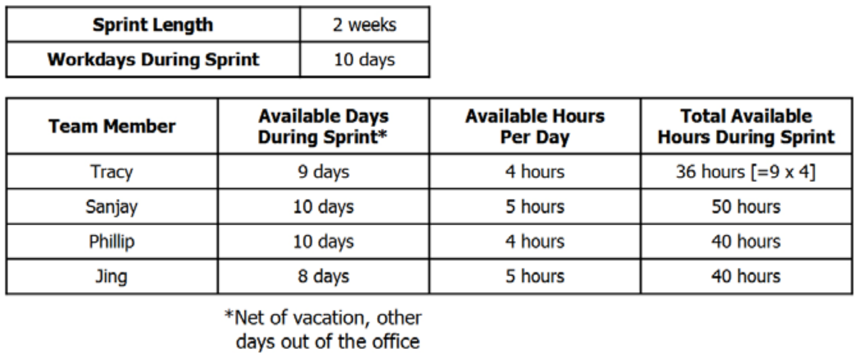
Figure 2. The Product Backlog

point during the project, the Product Backlog is the single, definitive view of "everything that needs to be done." Only a single Product Backlog exists; this means the Product Owner is required to make prioritization decisions across the entire spectrum of work to be done.

Items in the Product Backlog will vary widely in size; the larger ones will often be broken into smaller pieces during the Sprint Planning Meeting, and the smaller ones may be consolidated. One of the myths about Scrum is that it prevents you from writing detailed specifications; in reality, it's up to the Product Owner and Team to decide just how much detail is required, and this may vary from one Product Backlog item to the next. The general advice is to state what's important in the least amount of space necessary – in other words, one doesn't have to describe every possible detail of an item, one should just make clear what is necessary for it to be considered completed. The further down the Product Backlog one goes, the larger and less detailed the items will be; as they get closer to being worked on, additional detail gets filled in by the Product Owner.

Sprint Planning Meeting

At the beginning of each Sprint, the **Sprint Planning Meeting** takes place. In the first part of the Sprint Planning Meeting, the Product Owner and Scrum Team (with facilitation from the ScrumMaster) review the Product Backlog, discussing the goals and context for the items on the Backlog, and providing the Scrum Team with insight into the Product Owner's thinking. In the second part of the meeting, the Scrum Team selects the items from the Product Backlog to commit to complete by the end of the Sprint, starting at the top of the Product Backlog (in others words, starting with the items that are the highest priority for the Product Owner) and working down the list in order. This is one of the key practices in Scrum: the team decides how much work they will commit to complete, rather than having it assigned to them by the Product Owner. This makes for a much more reliable commitment; first, because the team is making it, rather than having it "made" for them by someone else; and second, because the team itself is determining how much work will be required, rather than having someone else decide how much "should" be required. While the Product Owner doesn't have any control over how much the team commits to, he or she knows that the items the team is committing to are drawn from the top of the Product Backlog – in other words, the items that he or she has rated as most important. The team does have the ability to pull in items from further down the list if it makes sense (for example, pulling in a slightly lower priority item that can be quickly completed as part of higher priority work).



Sprint Length	2 weeks
Workdays During Sprint	10 days

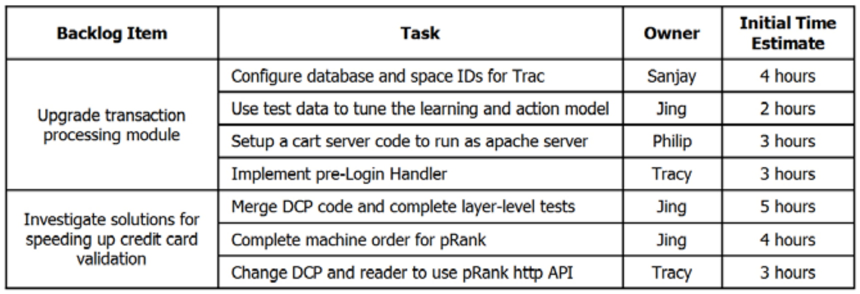
Team Member	Available Days During Sprint*	Available Hours Per Day	Total Available Hours During Sprint
Tracy	9 days	4 hours	36 hours [=9 x 4]
Sanjay	10 days	5 hours	50 hours
Phillip	10 days	4 hours	40 hours
Jing	8 days	5 hours	40 hours

*Net of vacation, other days out of the office

Figure 3. Estimating Available Hours

The Sprint Planning meeting will often last a number of hours – the team is making a very serious commitment to complete the work, and this commitment requires careful thought to be successful. The team will begin by estimating how much time each member has for Sprint-related work – in other words, their average workday minus the time they spend doing things like critical bug-fixes and other maintenance, attending meetings, doing email, taking lunch breaks, and so on. For most people this works out to 4-6 hours of time per day available for Sprint-related work. (Figure 3.)

Once the time available is determined, the team starts with the first item on the Product Backlog – in other words, the Product Owner's highest priority item – and working together, breaks it down into individual tasks, which are recorded in a document called the **Sprint Backlog** (figure 4). Once tasks are identified, team members will volunteer for them, thinking through dependencies and sequencing, making time estimates for each task, and making sure the workload of each individual is reasonable. There will be back and forth with the Product Owner during this process, to clarify points, verify tradeoffs, break down bigger Backlog items into smaller pieces, and generally ensure that the team fully understands what's being asked of it. The team will move sequentially down the Product Backlog in this way, until it's used up all its available hours. At the end of the meeting, the team will have produced a list of all the tasks, and for each task who has signed up to complete it and how much time they estimate it will take (typically in hours or fractions of a day). Many teams also make use of a visual task-tracking tool, in the form of a wall-sized task board where tasks (written on Post-It Notes) migrate during the Sprint across columns labeled "Not Yet Started", "In Progress", "To Verify", and "Completed".



Backlog Item	Task	Owner	Initial Time Estimate
Upgrade transaction processing module	Configure database and space IDs for Trac	Sanjay	4 hours
	Use test data to tune the learning and action model	Jing	2 hours
	Setup a cart server code to run as apache server	Philip	3 hours
	Implement pre-Login Handler	Tracy	3 hours
Investigate solutions for speeding up credit card validation	Merge DCP code and complete layer-level tests	Jing	5 hours
	Complete machine order for pRank	Jing	4 hours
	Change DCP and reader to use pRank http API	Tracy	3 hours

Figure 4. Sprint Backlog

One of the key pillars of Scrum is that once the Scrum Team makes its commitment, the Product Owner cannot add new requests during the course of the Sprint. This means that even if halfway through the Sprint the Product Owner decides that they want to add something new, he or she cannot make changes until the start of the next Sprint. If an external circumstance appears that significantly changes priorities, and means the team would be wasting its time if it continued working, the Product Owner can terminate the Sprint; this means the team stops all the work they are doing, and starts over with a Sprint Planning meeting, and so forth. The disruption of doing this is great, though, which serves as a disincentive for the Product Owner to resort to it except in extreme circumstances.

There is a powerful, positive influence that comes from the team being protected from changing goals during the Sprint. First, the team gets to work knowing with absolute certainty that its commitments will not change, which only reinforces the team's focus on ensuring completion. Second, it disciplines the Product Owner into really thinking through the items he or she prioritizes on the Product Backlog. Knowing that the commitment is for the duration of the Sprint makes the Product Owner much more diligent in deciding what to ask for at the beginning.

In return for all this, though, the Product Owner gets two things. First, he or she has the confidence of knowing the team has made a very strong commitment to complete the work they've signed up for, and over time Scrum teams get to be very good at delivering this. Second, the Product Owner gets to make whatever changes he or she likes to the Product Backlog before the start of the next Sprint. At this point, additions, deletions, modifications, and re-prioritizations are all completely acceptable. While the Product Owner is not able to make changes during the Sprint, he or she is always only a Sprint's duration or less away from making any changes whatsoever. Gone is the stigma around change – change of direction, change of requirements, or just plain changing your mind – and it may be for this reason that Product Owners are among the most ardent Scrum enthusiasts.

Daily Standup Meeting

Once the Sprint has started, the Scrum Team engages in another of the key Scrum practices: The **Daily Stand-Up Meeting**. This is a short (15 minute) meeting that happens every workday at an appointed time, and everyone on the Scrum Team attends; in order to ensure it stays brief, everyone stands (hence "Stand-Up Meeting"). It's the team's opportunity to report to itself on progress and obstacles. One by one, each member of the team reports just three things to the other members of the team: What they were able to get done since the last meeting, what they're aiming to get done by the next meeting, and any blocks or obstacles that are in their way. The ScrumMaster makes note of the blocks, and then helps team members to resolve them after the meeting. There's no discussion during the Daily Stand-Up Meeting, just the reporting of the three key pieces of information; if discussion is required, it takes place right after the meeting. The Product Owner, Managers, and other stakeholders can attend the meeting, but they should refrain from asking questions or opening discussion until after the meeting concludes – everyone should be clear that the team is reporting to each other, not to the Product Owner, Managers or ScrumMaster. While it's non-standard, some teams find it useful to have the Product Owners join and give a brief daily report of their own activities to the team.

After the meeting, the team members update the amount of time remaining to complete each of the tasks that they've signed up for on the Sprint Backlog. This information is recorded on a graph called the **Sprint Burndown Chart** (figure 5). It shows, each day, how much work (measured in hours or days) remains until the team's commitment is completed. Ideally, this should be a downward sloping graph that is on a trajectory to hit zero on the last day of the Sprint. And while sometimes it looks like that, often it doesn't. The important thing is that it show the team their actual progress towards their goal – and not in terms of how much time has been spent so far (an irrelevant fact, as far as Scrum is concerned), but in terms of how much work remains – what separates the team from their goal. If the curve is not tracking towards completion at the end of the Sprint, then the team needs to either pick up the pace, or simplify and pare down what it's doing. While this chart this can be maintained electronically

		Hours of Work Remaining on Each Day of the Sprint									
Task	Task Owner	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8	Day 9	Day 10
Configure database and space IDs for Trac	Sanjay	4	4	3	1	0					
Use test data to tune the learning and action model	Jing	2	2	2	2	1					
Setup a cart server code to run as apache server	Philip	3	3	5	2	0					
Implement pre-Login Handler	Tracy	3	3	3	3	3					
Merge DCP code and complete layer-level tests	Jing	5	5	2	2	2					
Complete machine order for pRank	Jing	4	4	3	3	3					
Change DCP and reader to use pRank http API	Tracy	3	3	0	0	0					
	Total	50	48	44	43	34					

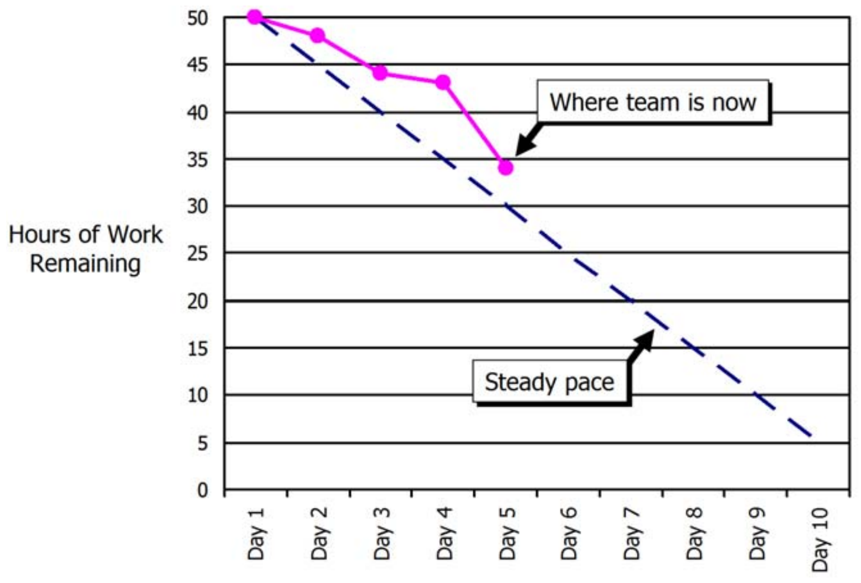


Figure 5. Daily Estimates and Burndown Chart

using Excel, many teams find it's easier and more effective to do it on paper taped to a wall in their workspace, with updates in pen; this low-tech solution is fast, simple, and often more visible than an electronic one.

One of the core tenets of Scrum is that the duration of the Sprint is never extended – it ends on the assigned date regardless of whether the team has completed the work it committed to or not. If the team has not completed their Sprint Goal, they have to stand up at the end of the Sprint and acknowledge that they did not meet their commitment. The idea is that this creates a very visible feedback loop, and teams are forced to get better at estimating what they are capable of accomplishing in a given Sprint, and then delivering it without fail. Teams will typically over-commit in their first few Sprints and fail to meet their Sprint Goal; they might then overcompensate and undercommit, and finish early; but by the third or fourth Sprint, teams will typically have figured out what they're capable of delivering, and they'll meet their Sprint goals reliably after that. Teams are encouraged to pick one duration for their Sprints (say, 2 weeks) and not change it frequently – a consistent duration helps the team learn how much it can accomplish, and it also helps the team achieve a rhythm for their work (this is often referred to as the "heartbeat" of the team).

Sprint Review

After the Sprint ends, there is the **Sprint Review**, where the team demos what they've built during the Sprint. Present at this meeting are the Product Owner, Team Members, and ScrumMaster, plus customers, stakeholders, experts, executives, and anyone else interested. This is not a "presentation" the team gives – there are no PowerPoints, and typically no more than 30 minutes is spent preparing for it – it's literally just a demo of what's been built, and anyone present is free to ask questions and give input. It can last 10 minutes, or it can last two hours – whatever it takes to show what's been built and get feedback.

Sprint Retrospective

Following the Sprint Review, the team gets together for the **Sprint Retrospective**. This is a practice that some teams skip, and that's unfortunate because it's one of the most important tools for making Scrum successful. It's an opportunity for the team to discuss what's working and what's not working, and agree on changes to try. The Scrum Team, the Product Owner, and the ScrumMaster will all attend, and a neutral outsider will facilitate the meeting; a good approach is for ScrumMasters to facilitate each others' retrospectives, which enables cross-pollination among teams.

A simple way to structure the Sprint Retrospective is to hang two sheets of poster-sized paper labeled "What's Working Well" and "What's Not Working, or Could Work Better" – and then have each person add several items to either. As items are repeated, check-marks are added next to them, so the common items become clear. Then the team looks for underlying causes, and agrees on changes to make in the upcoming Sprint, along with a commitment to review the results at the next Sprint Retrospective. Another useful practice is for the team to label each of the items in each column with either a "C" if it is caused by Scrum, or a "V" if it is made visible by Scrum (in other words, it would be happening with or without Scrum, but

Scrum makes it known to the team). The team may find a lot of C's on the "What's Working" side of the board, and a lot of V's on the "What's Not Working"; this is good news, even if the "What's Not Working" list is a long one, because the first step to solving underlying issues is making them visible, and Scrum is a powerful catalyst for that.

Starting the Next Sprint

Following the Sprint Review Meeting, the Product Owner takes all the input, as well as all new priorities that have appeared during the Sprint, and incorporates them into the Product Backlog; new items are added, and existing ones are modified, reordered, or deleted. Once this updating of the Product Backlog is complete, the cycle is ready to begin all over again, with the next Sprint Planning Meeting.

One practice many teams find useful is to hold a Prioritization Meeting toward the end of each Sprint, to review the Product Backlog for the upcoming Sprint with the Product Owner. In addition to giving the team an opportunity to suggest items the Product Owner may not be aware of – technical maintenance, for example – this meeting also kicks off any preliminary thinking that's required before the Sprint Planning Meeting.

There's no downtime between Sprints – teams will often go from a Sprint Review one afternoon into the next Sprint Planning Meeting the following morning. One of the values of Agile development is "sustainable pace", and only by working regular hours at a reasonable level of intensity can teams continue this cycle indefinitely.

Release Planning

Sprints continue until the Product Owner decides the product is almost ready for release, at which point there may be a "Release Sprint" to do final integration and testing in preparation for launch. If the team has followed good development practices along the way, with continuous refactoring and integration, and effective testing during each Sprint, there should be little cleanup required.

A question that's sometimes asked is how, in an iterative model, long-term release planning takes place. At the beginning of a project the team will do high-level release planning; since they cannot possibly know everything up front, the focus is on creating a rough plan to give the project broad direction, and clarify how tradeoff decisions will be made (scope versus schedule, for example). Think of this as the roadmap guiding you towards your final destination; which exact roads you take and the decisions you make during the journey will be determined en route.

Some releases are date-driven; for example: "We will release version 2.0 of our project at a trade-show on November 10." In this situation, the team will complete as many Sprints (and build as many features) as is possible in the time available. Other products require certain features to be built before they can be called complete and the product will not launch until these requirements are satisfied, however long that takes. Since Scrum emphasizes producing potentially shippable code each Sprint, teams may choose to start doing interim releases, to allow the customer to reap the benefits of completed work sooner.

Most Product Owners will choose one release approach, but inform it with the other – for example, they'll decide a release date, but they'll work with the team to come up with a rough estimate of the Backlog items that will be completed by that date. In situations where a "fixed price / fixed date / fixed deliverable" commitment is required – for example, contract development – at least one of those variables must have a built-in buffer to allow for uncertainty and change; in that respect, Scrum is no different from other development methodologies.

Common Challenges

Scrum tends to make visible a lot of issues that exist within the team, particularly at the beginning. For example, most teams are not good at estimating how much they can get done in a certain period, and so will fail to deliver what they committed to in the first Sprint. To the team, this feels like failure, and could cause them to question their adoption of Scrum. In reality, this experience is the necessary first step toward becoming better at estimating, and with the support of an experienced Scrum practitioner, the team can be helped to see it this way. Another difficulty a team might have is around the Daily Standup Meeting – getting all team members to commit to gather at a specified time, on time and every day without fail, may require the team to operate at a higher level than it's accustomed to, and some teams will be unable to do this.

One very common mistake teams make, when presented with a Scrum practice that challenges them, is to change the practice, not change themselves. For example, teams that have trouble delivering on their Sprint commitment might decide to make the Sprint duration extendable, so they never run out of time – and in the process, ensure they never have to learn how to do a better job of estimating and managing their time. In this way, without training and the support of an experienced Scrum coach, teams can morph Scrum into just a mirror image of their own weaknesses and dysfunction, and undermine the real benefit that Scrum offers: Making visible the good and the bad, and giving the team the choice of elevating itself to a higher level.

Another common mistake is to assume that a practice is discouraged or prohibited just because Scrum doesn't specifically require it. For example, Scrum doesn't specifically require the Product Owner to set a long-term strategy for his or her product; nor does it require engineers to seek advice from more experienced engineers (for example, their managers) about complex technical problems. Scrum leaves it to the individuals involved to make the right decision; and in most cases, both of these practices (along with many others) would be well-advised. To put it another way: "Just because Scrum doesn't say anything about breakfast doesn't mean you have to go hungry!"

Something else to be wary of is managers imposing Scrum on their teams; Scrum is about giving a team space and tools to self-organize, and having this dictated from above is not a recipe for success. A better approach might begin with a team learning about Scrum from a peer or manager, getting comprehensively educated in professional training, and then making a decision as a team to follow the practices faithfully for a defined period (say, 90 days); at the end of that period, the team will evaluate its experience, and decide whether to continue.

The good news is that while the first Sprint is often very challenging to the team, the benefits of Scrum tend to be visible by the end of it, leading many new Scrum teams to exclaim: "Scrum is hard, but it sure is a whole lot better than what we were doing before!"

Results From Scrum

The benefits of Scrum reported by teams come in various aspects of their experience. At Yahoo!, we have migrated nearly 50 projects to Scrum in the last 18 months, totaling almost 600 people, and the list of teams using it is quickly growing. These projects have ranged from consumer-facing, design-heavy websites like Yahoo! Photos, to the mission-critical back-end infrastructure of services like Yahoo! Mail, which serves hundreds of millions of customers; they range from entirely new products like Yahoo! Podcasts, which used Scrum from ideation through launch (and won a Webby Award for best product in its category that year), to more incremental projects, which included work on new features as well as bug fixes and other maintenance; and we've used Scrum for distributed projects, where the team is on separate continents. Once each quarter, we survey everyone at Yahoo! that is using Scrum (including Product Owners, Team Members, ScrumMasters, and the functional managers of those individuals) and ask them to compare Scrum to the approach they were using previously. An in-depth white paper on Yahoo!'s results is being prepared, but some early data is presented here:

- **Productivity:** 68% of respondents reported Scrum is better or much better (4 or 5 on a 5-point scale); 5% reported Scrum is worse or much worse (1 or 2 on a 5-point scale); 27% reported Scrum is about the same (3 on a 5-point scale).

- **Team Morale:** 52% of respondents reported Scrum is better or much better; 9% reported Scrum is worse or much worse; 39% reported Scrum is about the same.

- **Adaptability:** 63% of respondents reported Scrum is better or much better; 4% reported Scrum is worse or much worse; 33% reported Scrum is about the same.

- **Accountability:** 62% of respondents reported Scrum is better or much better; 6% reported Scrum is worse or much worse; 32% reported Scrum is about the same.

- **Collaboration and Cooperation:** 81% of respondents reported Scrum is better or much better; 1% reported Scrum is worse or much worse; 18% reported Scrum is about the same.

- **Team productivity increased an average a 36% increase, based on the estimates of the Product Owners.**

- **85% of team-members stated that they would continue using Scrum if the decision were solely up to them.**