This book is dedicated to Nobel Laureate Muhammad Yunus and the Grameen Bank for originating microenterprise development and the Accion International President's Advisory Board, responsible for much of microenterprise development in the western hemisphere.
The strategy for bootstrapping the poor out of poverty has been a model for freeing hundreds of thousands of software developers from developer abuse caused by poor management practices.

Thanks to the reviewers of the text who include among many others:

- Tom Poppendieck
- Henrik Kniberg
- Rowan Bunning
- Clifford Thompson
- Jim Coplien

About this book

This manual is based on The Scrum Papers, published by Scrum, Inc. For information on how to receive your own copy, please contact the author:

Jeff Sutherland
Scrum, Inc.
One Broadway, 14th Floor
Cambridge, MA 02142
Jeff.Sutherland@Scruminc.com

Executive Summary

Scrum is an agile method designed to add energy, focus, clarity, and transparency to project planning and implementation. Today, Scrum is used in small, mid-sized and large software corporations all over the world. It is being used in more and more areas beyond software.

Properly implemented, Scrum will:

- Increase speed of development
- Align individual and corporate objectives
- Create a culture driven by performance
- Support shareholder value creation
- Achieve stable and consistent communication of performance at all levels
- Enhance individual development and quality of life

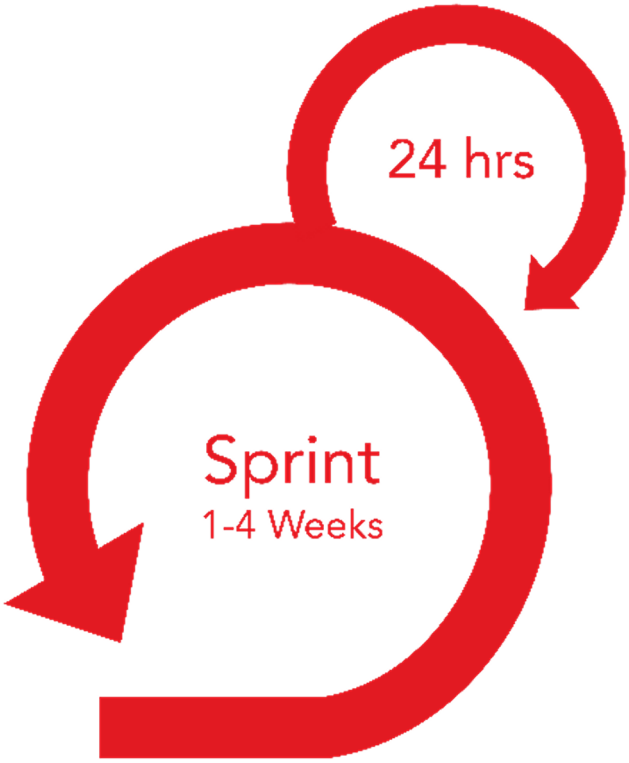
This handbook gives some basic information on how to get started with Scrum, and also describes some cases in point. It is based on The Scrum Papers, published by Scrum, Inc. (see www.scruminc.com).

Contents

In only twenty years…

Scrum has risen from being a method used by a number of enthusiasts at the Easel Corporation in 1993, to one of the world's most popular and well-known frameworks for development of software. The continued expansion of the global rollout of Scrum is testimony to the fact that Scrum delivers on its promise.

While it is often said that Scrum is not a silver bullet, Scrum can be like a heat-seeking missile when pointed in the right direction. Its inspect and adapt approach to continuous quality improvement can transform outmoded business practices. By focusing on building communities of stakeholders, encouraging a better life for developers, and delivering extreme business value to customers, Scrum can release creativity and team spirit in practitioners and make the world a better place to live and work.



Scrum has emerged from a rough structure for iterative, incremental development to a refined, well-structured, straight- forward framework for complex product development. I've worked with others to adjust, test, and adjust it again until it is solid. This framework is fully defined in the Scrum Guide at www.scrum.org, where Ken Schwaber and I sustain and help it emerge further.

The manual you are holding has been compiled from papers and compendiums that have been used at Scrum, Inc. ("The Scrum Papers"). We hope that it may serve both as an inspiration and a source of information for those readers who intend to start their first Scrum projects in their organizations. Seasoned Scrum users may also find some nuggets of wisdom. In any case, we appreciate all kinds of feedback. The Scrum adventure has just begun for all of us!

Chapter One

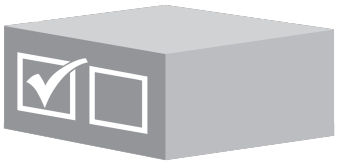
Scrum at a Glance

Scrum is an iterative, incremental framework for projects and product or application development.

Scrum structures development in cycles of work called Sprints. These iterations are less than one month in length, and usually measured in weeks. Sprints take place one after the other. The Sprints are of fixed duration – they end on a specific date whether the work has been completed or not, and are never extended. Hence, they are said to be time boxed.

At the beginning of each Sprint, a cross-functional team selects items (customer requirements) from a prioritized list. They commit to complete the items by the end of the Sprint. During the Sprint, the chosen items do not change. Every day the Team gathers briefly to re-plan its work to optimize the likelihood of meeting commitments.

At the end of the Sprint, the team reviews the Sprint with stakeholders, and demonstrates what they have built. People obtain feedback that can be incorporated in the next Sprint.



Inspect & adapt

Potentially Shippable Product Increment

Scrum emphasizes a working product at the end of the Sprint that is really "done". In the case of software, this means code that is:

- Integrated
- Fully Tested
- Potentially Shippable

A major theme in Scrum is "inspect and adapt." Since development inevitably involves learning, innovation, and surprises, Scrum emphasizes taking a short step of development, inspecting both the resulting product and the efficacy of current practices, and then adapting the product goals and process practices. Repeat forever.

Agile Development and Scrum

Scrum is, as the reader supposedly knows, an agile method. The agile family of development methods evolved from the old and well-known iterative and incremental life-cycle approaches. They were born out of a belief that an approach more grounded in human reality and the product development reality of learning, innovation, and change – would yield better results.

Agile principles emphasize building working software that people can get hands on quickly, versus spending a lot of time writing specifications up front. Agile development focuses on cross- functional teams empowered to make decisions, versus big

Scrum – A Rugby Term

"Scrum [---] in the sports of rugby union and rugby league, is a way of restarting the game, either after an accidental infringement or (in rugby league only) when the ball has gone out of play. [---] [A] scrum is formed by the players who are designated forwards binding together in three rows. The scrum then 'engages' with the opposition team so that the players' heads are interlocked with those of the other side's front row. The scrum half from the team that did not infringe then throws the ball into the tunnel created in the space between the two sets of front rowers' legs. Both teams may then try to compete for the ball by trying to hook the ball backwards with their feet."
(From Wikipedia)

hierarchies and compartmentalization by function. It also focuses on rapid iteration, with continuous customer input along the way. Often when people learn about agile development or Scrum, there's a glimmer of recognition – it sounds a lot like back in the start-up days "when we just did it."

Scrum was strongly influenced by a 1986 Harvard Business Review article on the practices associated with successful product development groups by Professors Takeuchi and Nonaka. in this paper the term "Scrum" was introduced, relating successful development to the game of Rugby in which a self-organizing (self- managing) team moves together down the field of product development. The first Scrum team was created at Easel Corporation in 1993 by Dr. Jeff Sutherland (the author of this manual) and the Scrum framework was formalized in 1995 by Jeff and Ken Schwaber.

Scrum's Reach

Today, Scrum is used by companies large and small, including:

Google, Yahoo!, Microsoft, Adobe
Lockheed Martin, Boeing, Raytheon
Johns Hopkins APL, Los Alamos Laboratories
Siemens, SAP, Oracle, IBM, Pegasystems
Nokia, Motorola, British Telecom, Telefonica/O2
Cisco, Alcatel, Ericsson
GE
Capital One, Wells Fargo, Vanguard, Saxo Bank
US Federal Reserve

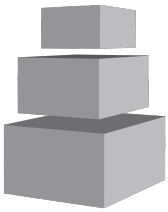
Teams using Scrum report significant improvements, and in some cases complete transformations, in both productivity and morale. For product developers – many of whom have been burned by the "management fad of the month club" – this is significant. Or to put it clearly: Scrum is just simple and powerful!

Part 1

Scrum Basics

Chapter 1

How Scrum Works



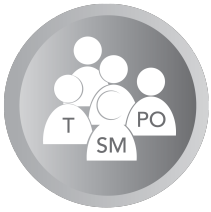
Product Backlog
(Features)

The Product Backlog

A Scrum project is driven by a product vision created by the Product Owner, and expressed in the Product Backlog. The Product Backlog is a prioritized list of what's required, ranked in order of value to the customer or business, with the highest value items at the top of the list. The Product Backlog evolves over the lifetime of the project, and items are continuously added, removed or reprioritized.

The Sprint

Scrum structures product development in cycles of work called Sprints, iterations of work that are typically 1–4 weeks in length. The Sprints are of fixed duration and end on a specific date whether the work has been completed or not; they are never extended.



Sprint Planning

Sprint Planning

At the beginning of each Sprint, the Sprint Planning Meeting takes place. The Product Owner and Team (with facilitation from the ScrumMaster) reviews the Product Backlog,

Three Roles:

Product Owner

Takes the inputs of what the product should be and translates them into a product vision or a Product Backlog.

The Team

Makes the product envisioned by the Product Owner.

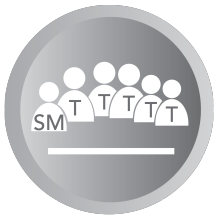
Scrum Master

Does whatever it takes to make the Scrum Team successful, such as removing organizational impediments, facilitating meetings, acting as a gatekeeper so no one unnecessary interrupts the team's work.

discuss the goals and context for the items, and the 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. Each item selected from the Product Backlog is designed and then broken down to a set of granulated steps. This list of backlog items is recorded in a document called the Sprint Backlog.

Daily Standup

Once the Sprint has started, the Team engages in another of the key Scrum practices: The Daily Stand-Up Meeting. This is a short (15 minutes) meeting that happens every workday at an appointed time. Everyone on the team attends. At this meeting, the information needed to inspect progress is presented. This information may result in re-planning and further discussions immediately after the Daily Standup.



Daily Standup

Sprint Review

Sprint Review

After the Sprint ends, there is the Sprint Review, where the Scrum Team and stakeholders inspect what was done during the Sprint, discuss it, and figure out what to do next. Present at this meeting are the Product Owner, Team Members, and ScrumMaster, plus customers, stakeholders, experts, executives, and anyone else interested.

Sprint Retrospective

Following the Sprint Review, the team gets together for the Sprint Retrospective which is an opportunity for the team to discuss what's working and what's not working, and agree on changes to try.

Sprint Retrospective

What's Wrong With Traditional Software Development?

The traditional way to build software, used by companies big and small, was a sequential life cycle of which there are many variants (such as the V-Model). Commonly, it is known as "The Waterfall".

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 organized using tools such as Gantt charts and applications such as Microsoft Project. The team arrives at an estimate of how long the development 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 work.

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 testing organization (some call this Quality Assurance), 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. Hence a lot of problems occur:

Creativity Is Inhibited

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

Written Documents Have Limitations

The waterfall approach 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 50-page requirements documents just do not get read. When they do get read, the misunderstandings are often compounded. A written document is an incomplete picture of my ideas; when you read it, you create another abstraction, which is now two steps away from what I think I meant to say at that time. It is no surprise that serious misunderstandings occur.

Bad Timing

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. You immediately think of 20 ways you could have made it better. Unfortunately, these very valuable insights often come at the end of the release cycle, when changes are most difficult and disruptive – in other words, when doing the right thing is most expensive, at least when using a traditional method.

No Crystal Balls

Humans are not able to predict the future. For example, your competition makes an announcement that was not expected. Unanticipated technical problems crop up that force a change in direction. Furthermore, people are particularly bad at planning uncertain things far into the future – guessing today how you will be spending your week eight months from now is something of a fantasy. It has been the downfall of many a carefully constructed Gantt chart.

Too Much Work and No Fun

In addition, a sequential life cycle tends to foster an adversarial relationship between the people that are handing work off from one to the next. "He's asking me to build something that's not in the specification." "She's changing her mind." "I can't be held responsible for something I don't control." And this gets us to another observation about sequential development – it is not much fun. The waterfall model is a cause of great misery for the people who build products. The resulting products fall well short of expressing the creativity, skill, and passion of their creators. People are not robots, and a process that requires them to act like robots results in unhappiness.

Sub-optimized results

A rigid, change-resistant process produces mediocre products. Customers may get what they first ask for (at least two translation steps removed), but is it what they really want once they see the product? By gathering all the requirements up front and having them set in stone, the product is condemned to be only as good as the initial idea, instead of being the best once people have learned or discovered new things.

Practitioners of a sequential life cycle experience these shortcomings again and again. But, it seems so supremely logical that the common reaction is to turn inward: "If only we did it better, it would work, and 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! There are also management teams that have invested their reputation – and many resources – in a waterfall model; changing to a fundamentally different model is an apparent admission of having made a mistake. And Scrum is fundamentally different ...

Chapter 2

The Scrum Roles

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

Product Owner

The Product Owner

The Product Owner is responsible for maximizing return on investment (ROI) by identifying product features, translating these into a prioritized feature list, deciding which should be at the top of the list for the next Sprint, and continually re- prioritizing and refining the list.

The Product Owner has profit and loss responsibility for the product, assuming it is a commercial product. In the case of an internal application, the Product Owner is not responsible for ROI in the sense of a commercial product (that will generate revenue), but they are still responsible for maximizing ROI in the sense of choosing – each Sprint – the highest- business-value lowest-cost items.

Not a Product Manager

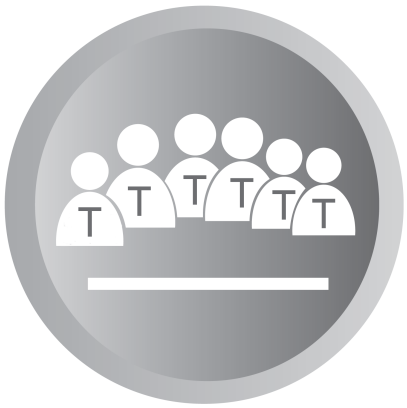
In some cases, the Product Owner and the customer are the same person; this is common for internal applications. In others, the customer might be millions of people with a variety of needs, in which

case the Product Owner role is similar to the Product Manager or Product Marketing Manager position in many product organizations.

However, the Product Owner is somewhat different than a traditional Product Manager because they actively and frequently interact with the team, personally offering the priorities and reviewing the results each two- or four-week iteration, rather than delegating development decisions to a project manager. It is important to note that in Scrum there is one and only one person who serves as – and has the final authority of – Product Owner. In multi- team programs, this one Product Owner may delegate the work to Product Owners that represent him or her on subordinate teams, but all decisions and direction come from the top-level, single Product Owner.

The Team

The Team builds the product that the customer is going to use: the application or website, for example. The Scrum team is cross-functional and includes all the expertise necessary to deliver the potentially shippable product each Sprint. It is also self-organizing (self-managing), with a very high degree of autonomy and accountability.



The Team

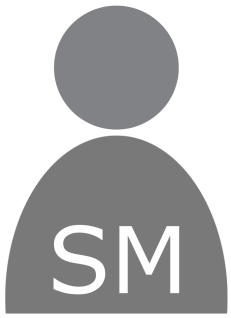
Hence, there is no team manager or project manager in Scrum. Instead, the Team members decide what to commit to, and how best to accomplish that commitment. The Team is self- organizing.

The Scrum Team includes the Product Owner and the Scrum Master. However, the Team often refers to those implementing the Sprint Backlog, which may or may not include the Product Owner or the Scrum Master.

Dedicated Team

The Scrum Team is seven plus or minus two people. For a software product the Team working on the Sprint Backlog might include programmers, interface designers, and testers. The Team develops the product and provides ideas to the Product Owner about how to make the product great. In my experience, it is essential that the Team is 100 percent dedicated to the work for one product during the Sprint; multitasking across multiple products or projects will severely limit performance.

Stable Teams are associated with higher productivity, so changing team members should also be avoided. Application groups with many people are organized into multiple Scrum teams, each focused on different features for the product, with close coordination of their efforts. Since one Team does all the work (planning, analysis, programming, and test) for a complete customer-centric feature, Scrum teams are also known as feature teams. In very technically complex programs and products, I've seen Teams organized by architectural layer - such as when product family architectures are employed. However, integration prior to the end of the Sprint is more difficult when Teams are so structured.



The Scrum Master

The Scrum Master helps the product group learn and apply Scrum to achieve business value. The Scrum Master does whatever is in their power to help the team be successful.

The Scrum Master is not the manager of the team or a project manager; instead, the Scrum Master serves the team, protects them from outside interference, and educates and guides the Product Owner and the team in the skillful use of Scrum. The Scrum Master makes sure everyone on the team (including the Product Owner, and those in management) understands and follows the practices of Scrum. They also help lead the organization through the often difficult changes required to achieve success with agile development.

Commitment is Important

Since Scrum makes visible many impediments and threats to the team's and Product Owner's effectiveness, it is important to have an engaged Scrum Master working energetically to help resolve those issues. If not, the team or Product Owner will find it difficult to succeed. Scrum teams should have a dedicated full-time Scrum Master, although a smaller team might have a team member play this role (carrying a lighter load of regular work when they do so). Great Scrum Masters can come from any background or discipline: Engineering, Design, Testing, Product Management, Project Management, or Quality Management.

The Scrum Master and the Product Owner cannot be the same individual; at times, the Scrum Master may be called upon to push back on the Product Owner (for example, if they try to introduce new

deliverables in the middle of a Sprint). And unlike a project manager, the Scrum Master does not tell people what to do or assign tasks – they facilitate the process, supporting the team as it organizes and manages itself. If the Scrum Master was previously in a position managing the team, they will need to significantly change their mindset and style of interaction for the team to be successful with Scrum. In the case that an ex-manager transitions to the role of Scrum Master, it is best to serve a team other than the one that they used to supervise.

What About Managers?

Note that there is no role of project manager in Scrum. Sometimes an (ex-) project manager can step into the role of Scrum Master, but this has a mixed record of success – there is a fundamental difference between the two roles, both in day-to-day responsibilities and in the mindset required to be successful.

In addition to the three Scrum roles, there are other contributors to the success of the product, including managers. While their role changes, they are invaluable. For example:

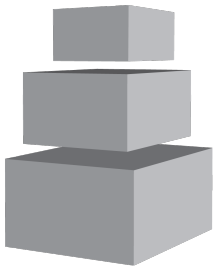
- They create a business model that works and provide resources the team needs
- They support the team by respecting the rules and spirit of Scrum
- They help remove impediments that the team identifies
- They make their expertise and experience available to the team
- They challenge the team to move beyond mediocrity

In Scrum, these individuals replace the time they previously spent playing the role of "nanny" (assigning tasks, getting status reports, and other forms of micromanagement) with time as "guru" and "servant" of the team (mentoring, coaching, helping remove obstacles, helping problem-solve, providing creative input, and guiding the skills development of team members). In this shift, managers may need to change 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.

Chapter 3

Getting Started

Initiating a Scrum project is not hard, as long as one takes one step at a time, and makes sure that everyone feels included.



The Product Backlog

The first step in Scrum is for the Product Owner to articulate the product vision. Eventually, this evolves into the refined and prioritized list of features, the Product Backlog.

Product Backlog (Features)

This backlog exists and evolves over the lifetime of the product; it is the product road map. At any point, the Product Backlog is the single, definitive view of "everything that could be done by the team ever, in order of priority". Only a single Product Backlog exists; this means the Product Owner is required to make prioritization decisions across the entire spectrum.

How Much Detail?
One of the myths about Scrum is that it prevents you from writing detailed specifications; in reality, it is up to the Product Owner and Team to decide how much detail is required, and this will vary from one backlog item to the next, depending on the insight of the team, and other factors. State what is important in the least amount of space necessary – in other words, do not describe every possible detail of an item, just make clear what is necessary for it to be understood. Low priority items, which are likely to be implemented at a later stage and are usually "coarse–grained", have fewer requirement details. High priority and "fine-grained items" that will soon be implemented tend to have more detail. For more on structuring Product Backlog, a study of lean thinking, particularly lean inventory and just-in-time order processing, will prove instructional.

The Product Backlog includes a variety of items, primarily new customer features ("enable all users to place book in shopping cart"), but also engineering improvement goals ("rework the transaction processing module to make it scalable"), exploratory or research work ("investigate solutions for speeding up credit card validation"), performance and security requirements, and, possibly, known defects ("diagnose and fix the order processing script errors"), if there are only a few problems. (A system with many defects usually has a separate defect tracking system.) Many people like to articulate the requirements in terms of "user stories" - concise, clear descriptions of the functionality in terms of its value to the end user of the product. In more demanding environments, such as FDA life critical applications, Use Cases are often used.

The subset of the Product Backlog that is intended for the current release is known as the Release Backlog, and in general, this portion is the primary focus of the Product Owner.

					New Estimates of Effort Remaining as of Sprint...					
Item	**Details (wiki URL)**	**Priority**	**Estimate of Value**	**Initial Estimate of Effort**	**1**	**2**	**3**	**4**	**5**	**6**
As a buyer, I want to place a book in a shopping cart (see UI sketches on wiki page)	...	1	7	5						
As a buyer, I want to remove a book in a shopping cart	...	2	6	2						
Improve transaction processing performance (see target performance metrics on wiki)	...	3	6	13						
Investigate solutions for speeding up credit card validation (see target performance metrics on wiki)	...	4	6	20						
Upgrade all servers to Apache 2.2.3	...	5	5	13						
Diagnose and fix the order processing script errors (bugzilla ID 14823)	...	6	2	3						
As a shopper, I want to create and save a wish list	...	7	7	40						
As a shopper, I want to to add or delete items on my wish list	...	8	4	20						

The Product Backlog leads the way ahead for the Scrum Team. Maintained by Product Owner.

The Product Backlog is continuously updated by the Product Owner to reflect changes in the needs of the customer, new ideas or insights, moves by the competition, technical hurdles that appear, and so forth. The team provides the Product Owner with estimates of the effort required for each item on the Product Backlog. In addition, the Product Owner is responsible for assigning a business value estimate to each individual item. This is often an unfamiliar practice for a Product Owner. With the two estimates (effort and value) and perhaps with additional risk estimates, the Product Owner prioritizes the backlog (actually, usually just the Release Backlog subset) to maximize ROI (choosing items of high value with low effort) or secondarily, to reduce some major risk. As will be seen, these effort and value estimates may be refreshed each Sprint as people learn; consequently, this is a continuous re-prioritization activity and the Product Backlog is ever evolving.

Scrum does not mandate the form of estimates in the Product Backlog, but it is common to use relative estimates expressed as “points” rather than absolute units of effort such as person-weeks.

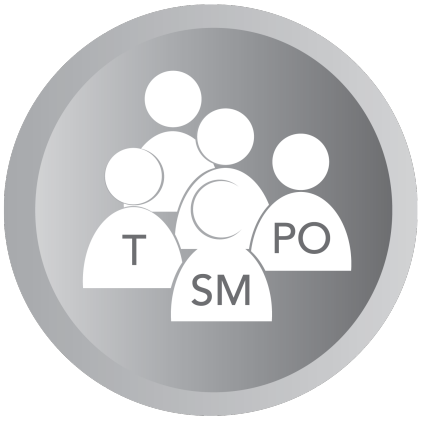
Team Planning
A key practice in Scrum is that the team decides how much work they will commit to complete, rather than having it assigned to them by the Product Owner. This makes for a more reliable commitment because the team is making it based on their own analysis and planning, rather than having it "made" for them by someone else.

Over time, a team tracks how many relative points they implement each Sprint; for example, averaging 26 points per Sprint. With this information they can project a release date to complete all features, or how many features will likely be completed by a date. Standard deviations around the average points will indicate least likely and most likely possibilities. The number of points completed per Sprint is called the velocity of the team. A realistic release plan is always based on the velocity of the team.

The items in the Product Backlog can vary significantly in size or effort. Larger ones are broken into smaller items during the Product Backlog Refinement workshop or the Sprint Planning Meeting, and smaller ones may be consolidated.

Sprint Planning

The Sprint Planning Meeting opens the Sprint. It is divided into two distinct sub-meetings, the first of which is called Sprint Planning Part One.



Sprint Planning

In Sprint Planning Part One, the Product Owner and Team (with facilitation from the Scrum Master) review the high-priority items in the Product Backlog that the Product Owner is interested in

implementing this Sprint. They discuss the goals and context for these high-priority items on the Product Backlog, providing the Team with insight into the Product Owner's thinking. The Product Owner and Team also review the "Definition of Done" that all items must meet, such as, "Done means coded to standards, reviewed, implemented with unit test-driven development (TDD), tested with 100 percent test automation, integrated, and documented." This definition of "done" ensures transparency and quality fit for the purpose of the product and organization.

Part One focuses on understanding what the Product Owner wants. According to the rules of Scrum, at the end of Part One the (always busy) Product Owner may leave although they must be available (for example, by phone) during the next meeting. However, they are welcome to attend Part Two...

Sprint Planning Part Two, often referred to as Sprint Refinement, focuses on detailed task planning for how to implement the items that the team decides to take on. The Team selects the items from the Product Backlog they 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.

While the Product Owner does not have 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 has the authority to also select items from further down the list in consultation with the Product Owner; this usually happens when

the team and Product Owner realize that something of lower priority fits easily and appropriately with the high priority items.

One Item at a Time

During task generation and estimation in Sprint Planning it is not necessary – nor appropriate – for Team members to volunteer for all the tasks "they can do best." Rather, it is better to only volunteer for one task at a time, when it is time to pick up a new task and to consider deliberately choosing tasks that will involve learning (perhaps by pair work with a specialist).

The Sprint Planning Meeting should be time boxed to four hours for a four-week Sprint and two hours for a two-week Sprint. In order to do this, the team must help the Product Owner by estimating the size of stories before the Sprint Planning meeting – the team is making a serious commitment to complete the work, and this commitment requires careful thought to be successful. A Team bases its commitments on its past velocities. If a Team is new, new to the technology or domain, it may not have reliable, stable velocities until it has worked together for three or four Sprints. In making its commitment, the Team factors in any vacations, new organizational demands, and other items that may reduce its past velocity.

Once the Team capacity 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 stories, which are recorded in a document called the Sprint Backlog (see below). As mentioned, the Product Owner must be available during Part Two (such as via the phone) so that clarifications and decisions regarding alternative approaches is possible. The team will move sequentially down the Product Backlog in this way, until it's used up all its capacity. At the end of the meeting, the team will have produced a list of tasks with estimates (typically in hours or fractions

of a day). The list is a starting point, but more tasks will emerge as the Team addresses each Product Backlog item during the Sprint. The Team will work on a technical design that will be implemented using Sprint Backlog tasks. The team choses the ordering of Sprint Backlog tasks to maximize the velocity of production and quality of "done" functionality.

Scrum encourages multi-skilled workers, rather than only "working to job title" such as a "tester" only doing testing. In other words, team members "go to where the work is" and help out as possible. If there are many testing tasks, then all Team members may help. This does not imply that everyone is a generalist; no doubt some people are especially skilled in testing (and so on) but Team members work together and learn new skills from each other. Pairing has proven a valuable approach to sharing knowledge.

All that said, there are rare times when a Team member may do a particular task because it would take far too long or be impossible for others to learn – perhaps he or she is the only person with the artistic skill to draw pictures. Other Team members could not draw a "stick man" if their lives depended on it. In this rare case – and if it is not rare and not getting rarer as the Team learns, there is something wrong – it may be necessary to ask if the total planned drawing tasks that must be done by this certain Team member are feasible within the short

No Changing Goals

There are powerful, positive factors that arise 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, thus reinforcing 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 and offers to the team for the Sprint.

Sprint.

One of the pillars of Scrum is that once the Team makes its commitment, any additions or changes must be deferred until the next Sprint. This means that if halfway through the Sprint the Product Owner decides there is a new item he or she would like the Team to work on, he cannot make the change 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 or the team can terminate the Sprint. The Team stops, and a new Sprint Planning meeting initiates a new Sprint. The disruption of doing this is usually great; this serves as a disincentive for the Product Owner or team to resort to this dramatic decision.

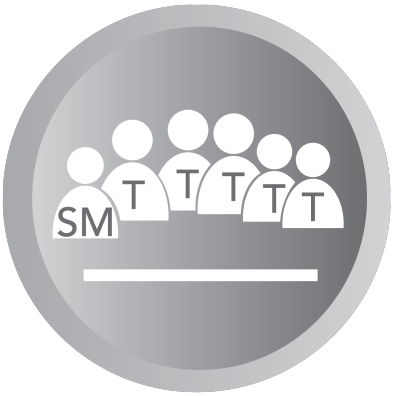
Many teams also make use of a visual task-tracking tool, in the form of a wall-sized task board where tasks (written on Post-It Notes) migrate during the Sprint across columns labeled:

"To Do," "Doing," and "Done."

By following these Scrum rules the Product Owner gains two things. First, he or she has the confidence of knowing the Team has made a commitment to complete a realistic and clear set of tasks they have chosen. Over time a Team can become quite skilled at choosing and delivering on a realistic commitment. 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 that point, additions, deletions, modifications, and re-prioritizations are all possible and acceptable. While the Product Owner is not able to make changes to the selected items under development during the current Sprint, he or she is only one Sprint's duration or less away from making any changes. 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 usually as enthusiastic about Scrum as anyone.

Daily Standup



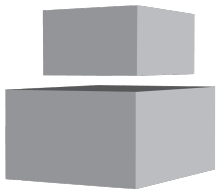
Once the Sprint has started, the Team engages in another of the key Scrum practices: The Daily Standup. This is a short (15 minutes or less) meeting that happens every workday at an appointed time and place. Everyone on the Team attends. To keep it brief, it is recommended that everyone remain standing. It is the Team's opportunity to talk to each other and inspect each other's progress and obstacles. In the Daily Standup, one by one, each member of the team reports three (and only three) things to the other members of the team:

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

Note that the Daily Standup is not a status meeting or a report to a manager; it is a time for a self-organizing Team to share with each other what is going on, to help it coordinate its work and optimize its likelihood of meeting its commitments. Someone makes note of the blocks, and the Scrum Master is responsible for helping team members resolve them.

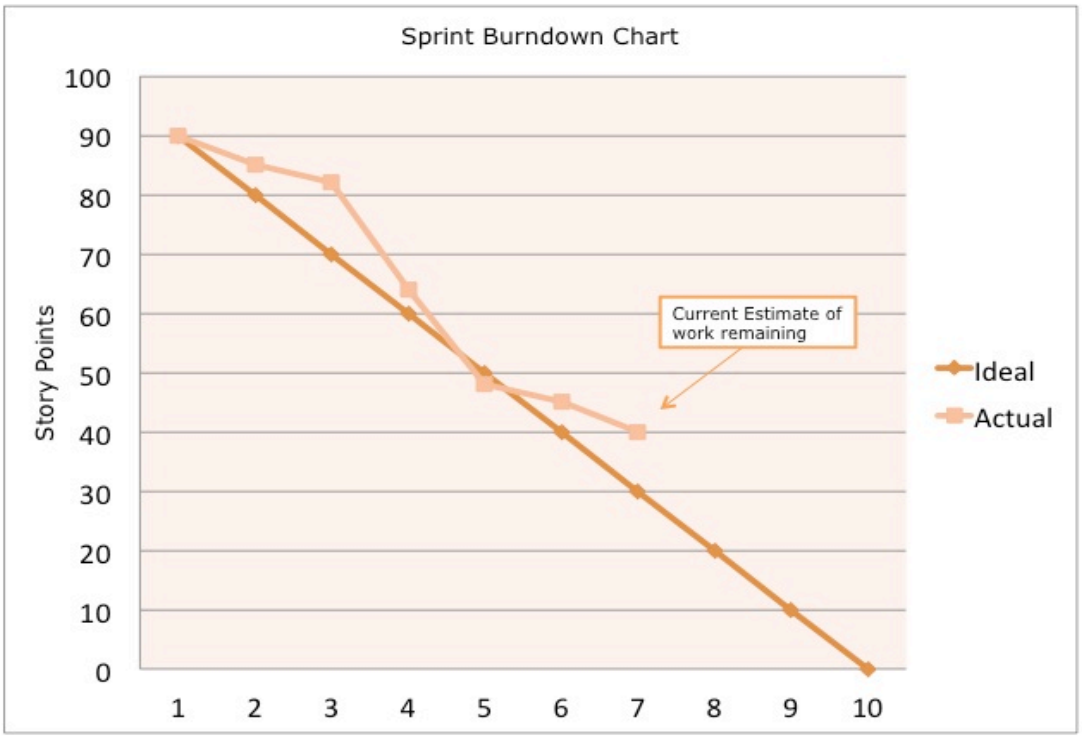
There is no chit-chat during the Daily Standup, only reporting answers to the three questions; if discussion is required it takes place immediately after the Daily Standup in a follow-up meeting, although in Scrum no one is required to attend this. This follow-up meeting is a common event where the Team adapts to the information they heard in the Daily Standup: in other words, another inspect and adapt cycle. It is generally recommended not to have managers or others in positions of perceived authority attend the Daily Standup. This risks making the Team feel "monitored" – under pressure to report major progress every day (an unrealistic expectation), and inhibited about reporting problems – and it tends to undermine the Team's self-management, and invite micromanagement. It would be more useful for a stakeholder to instead reach out to the team following the meeting, and offer to help with any blocks that are slowing the Team's progress.

Updating Sprint Backlog & Sprint Burndown Chart



Sprint Backlog (Stories)

Every day, the Team members update their estimate of the amount of time remaining to complete their current task in the Sprint Backlog. Following this update, someone adds up the points remaining for the Team as a whole, and plots it on the Sprint Burndown Chart. This graph shows, each day, a new estimate of how much work (measured in relative points) remains until the Team's tasks are finished. Ideally, this is a downward sloping graph that is on a trajectory to reach "zero effort remaining" by the last day of the Sprint. Hence it is called a burndown chart. And while sometimes it looks good, often it does not; this is the reality of product development. The important thing is that it shows the Team their progress towards their goal, not in terms of how much time was spent in the past (an irrelevant fact in terms of progress), but in terms of how much work remains in the future – what separates the Team from their goal.



Sprint Burndown Chart. While the Sprint Burndown chart can be created and displayed using a spreadsheet, many teams find it is more effective to show it on paper on a wall in their workspace, with updates in pen; this "low-tech/high-touch" solution is fast, simple, and often more visible than a computer chart.

If the burndown line is not tracking downwards towards completion by mid-Sprint, the team needs to execute the Scrum Emergency Procedure:

1. Change the approach to the work or remove impediments to increase velocity.
2. Get help by having someone outside the team take some of the backlog.
3. Reduce the scope of work.

4. Abort the Sprint.

It is important that the Scrum Master coach the Team to take action early rather than drifting into Sprint failure. Some Scrum Masters insist that a Team reduce its commitments in early Sprints. Successful Teams consistently improve by building on success. Failing Teams stay stuck at low velocity.

Product Backlog Refinement

Product Backlog Refinement

One of the lesser known, but valuable, guidelines in Scrum is that five or ten percent of each Sprint must be dedicated by the Product Owner and the team to refining the Product Backlog. This includes:

- Detailed requirements analysis
- Splitting large items into smaller ones
- Estimation of new items
- Re-estimation of existing items

A regularly scheduled Weekly meeting with the Product Owner is enough for experienced Teams to refine the Product Backlog. This refinement activity is not for items selected for the current Sprint; it is for items for the future, most likely in the next one or two Sprints. With this practice, Sprint Planning becomes relatively simple because the Product Owner and Scrum Team start the planning with a clear, well analyzed and carefully estimated set of items. A sign that this refinement process is not being done (or not being done well) is that Sprint Planning involves significant questions, discovery, or confusion.

Ending the Sprint

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. Teams typically over-commit in the first few Sprints and fail to meet objectives. Teams might then overcompensate and under-commit, and finish early. But by the third or fourth Sprint, a Team typically has figured out what it are capable of delivering (most of the time), and it will meet its Sprint goals more reliably after that. Teams are encouraged to pick one duration for Sprints (say, two weeks) and not change it. A consistent duration helps the Team learn how much it can accomplish, which helps in both estimation and longer-term release planning. It also helps the Team achieve a rhythm for their work; this is often referred to as the "heartbeat" of the team in Scrum.

Sprint Review

After the Sprint ends, there is the Sprint Review, where the team reviews the Sprint with the Product Owner. This is often mislabeled the "demo" but that does not capture the real intent of this meeting. A key idea in Scrum is inspect and adapt. To see and learn what is going on and then evolve based on feedback, in repeating cycles. The Sprint Review is an inspect and adapt activity for the product. It is a time for the

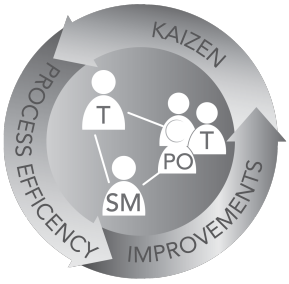
Sprint Review

Product Owner and key stakeholders to learn what is going on with the product and with the Team (that is, a review of the Sprint); and for the Team to learn what is going on with the Product Owner and the market. Consequently, the most important element of the Review is an in-depth conversation and collaboration between the Team and Product Owner to learn the situation, to get advice, and so forth. The review includes a demo of what the Team built during the Sprint, but if the focus of the review is a demo rather than conversation, there is an imbalance.

Present at this meeting are the Product Owner, Team members, and Scrum Master, plus customers, stakeholders, experts, executives, and anyone else interested. The demo portion of the Sprint Review is not a "presentation" the team gives – there is no slideware. A guideline in Scrum is that as little time as possible should be spent on preparing for the Sprint Review; Scrum suggests no more than 2 hours. It is simply a demo of what has been built. Anyone present is free to ask questions and give input.

The Sprint Retrospective

The Sprint Review involves inspect and adapt regarding the product. The Sprint Retrospective, which follows the Review, involves inspect and adapt regarding the process. This is a practice that some teams skip which is unacceptable, because self-organization requires the frequent regular reflection provided by the Retrospective. It's the main mechanism for taking the visibility that Scrum provides into areas of potential improvement, and turning it into results. It's an opportunity for the entire Scrum Team to discuss what's working and what's not working, and agree on changes to try. Sometimes the



Sprint Retrospective

One Way To Do It....

A simple way to structure the Sprint Retrospective is to draw four columns on a whiteboard, labeled:

- What went well?
- What could have been better?
- Things to try?
- Issues to escalate?

Then go around the room, with each person adding one or more items to the lists. 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 a small number of changes to try in the upcoming Sprint, along with a commitment to review the results at the next Sprint Retrospective.

A useful practice at the end of the Retrospective is for the team to label each of the items in each column with either a "C" if it is caused by Scrum (in other words, without Scrum it would not be happening), or an "E" if it is exposed by Scrum (in other words, it would be happening with or without Scrum, but Scrum makes it known to the team), or a "U" if it's unrelated to Scrum (like the weather). The team may find a lot of C's on the "What's Working Well" side of the board, and a lot of E's on the "What Could Work Better "; this is good news, even if the "What Could Work Better" list is a long one, because the first step to solving underlying issues is making them visible,

Scrum Master can act as an effective facilitator for the retrospective, but it may be better to find a neutral outsider to facilitate the meeting; a good approach is for Scrum Masters to facilitate each others' retrospectives, which enables cross-pollination among teams.

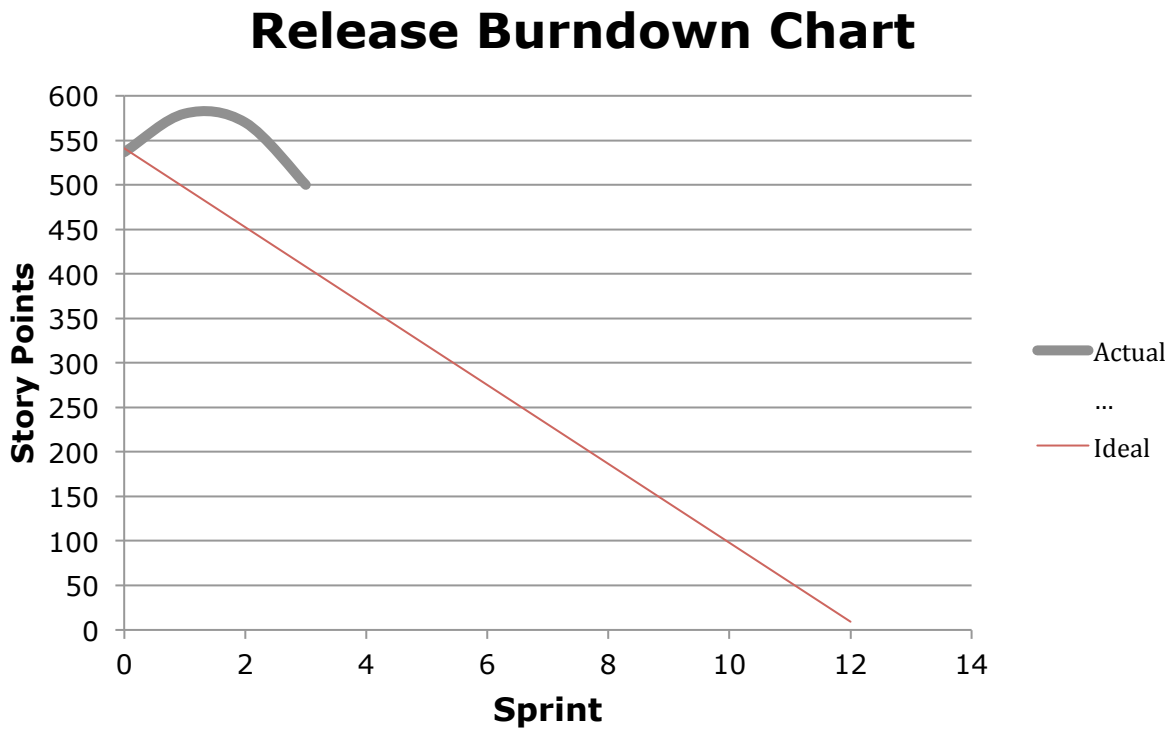
Updating Release Backlog & Burndown Chart

At this point, some items have been finished, some have been added, some have new estimates, and some have been dropped from the release goal. The Product Owner is responsible for ensuring that these changes are reflected in the Release Backlog (and more broadly, the Product Backlog). In addition, Scrum includes a Release Burndown chart that shows progress towards the release date. It is analogous to the Sprint Burndown chart, but is at the higher level of items (requirements) rather than fine-grained tasks. Since a new Product Owner is unlikely to know why or how to create this chart, this is another opportunity for a Scrum Master to help the Product Owner.

	Priority	Estimate of value	Estimate of Effort	1	2	3
As a user I want to put a book in my cart (see UI sketches on wiki page)	1	7	5	0	0	0
As a buyer I want to remove a book from my shopping cart.	2	6	7	0	0	0
Improve transaction processing performance (see metrics on wiki)	3	6	13	13	0	0
Investigate solutions for speeding up credit card transactions (see metrics on wiki.	4	6	20	20	20	0

Upgrade all servers to latest version of Apache	5	5	13	13	13	13
Diagnose and fix order processing script errors (bugzilla ID 48133)	6	2	3	3	3	3
As a buyer I want to add items to my wish list	7	7	40	40	40	40
As a buyer I want to delete items on my wish list	8	4	20	20	20	20
And so on…	…	…	…	…	…	…
New estimate of effort remaining at the end of each Sprint		Total	537	580	570	500

Release Backlog (a subset of the Product Backlog)



Starting the Next Sprint

Following the Sprint Review, the Product Owner may update the Product Backlog with any new insights. At this point, the Product Owner and Team are ready to begin another Sprint cycle. There is no down time between Sprints – teams normally go from a Sprint Retrospective one afternoon into the next Sprint Planning the following morning (or after the weekend).

One of the principles of agile development is "sustainable pace", and only by working regular hours at a reasonable level can teams continue this cycle indefinitely.

Release Sprint

The perfection vision of Scrum is that the product is potentially shippable at the end of each Sprint, which implies there is no wrap up work required, such as testing or documentation. Rather, the implication is that everything is completely finished every Sprint; that you could actually ship it or deploy it immediately after the Sprint Review.

However, many organizations have weak development practices and cannot achieve this perfection, or there are other extenuating circumstances (such as, "the machine broke"). In this case, there will be some remaining work, such as final production environment integration testing, and so there will be the need for a "Release Sprint" to handle this remaining work. A goal of any Scrum Team is to minimize the number of Release Sprints for completing "undone" work. Undone work tends to accumulate exponentially and causes poor product quality.

Release Planning & Initial Product Backlog Refinement

A question that is sometimes asked is how, in an iterative model, can long-term release planning be done. There are two cases to consider:

1. A new product in its first release
2. An existing product in a later release

In the case of a new product, or an existing product just adopting Scrum, there is the need to do initial Product Backlog refinement before the first Sprint, where the Product Owner and team shape a proper Scrum Product Backlog. This could take a few days or a week, and involves a vision workshop, some detailed requirements analysis, and estimation of all the items identified for the first release.

> **"Change Thyself!"**
> One common mistake teams make, when presented with a Scrum practice that challenges them, is to change Scrum, 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 coaching and the support of an experienced Scrum Master, organizations can mutate Scrum into just a mirror image of its own weaknesses and dysfunction, and undermine the real benefit that Scrum offers: making visible the good and the bad, and giving the organization the choice of elevating itself to a higher level.

Surprisingly in Scrum, in the case of an established product with an established Product Backlog, there should not be the need for any special or extensive release planning for the next release. Why? Because the Product Owner and team should be doing Product Backlog refinement every Sprint (five or ten percent of each Sprint),

continuously preparing for the future. This continuous product development mode obviates the need for the dramatic punctuated prepare-execute-conclude stages one sees in traditional sequential life cycle development.

During an initial Product Backlog refinement workshop and during the continuous backlog refinement each Sprint, the Team and Product Owner will do release planning, refining the estimates, priorities, and content as they learn.

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, the Product Owner may choose to start doing interim releases, to allow the customer to reap the benefits of completed work sooner.

Since they cannot possibly know everything up front, the focus is on creating and refining a plan to give the release 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 destinations; which exact roads you take and the decisions you make during the journey may be determined en route.

Most Product Owners choose one release approach. For example, they will decide a release date, and will work with the team to estimate the Release Backlog items that can be completed by that date. In

situations where a "fixed price / fixed date / fixed deliverable" commitment is required – for example, contract development – one or more of those parameters must have a built-in buffer to allow for uncertainty and change; in this respect, Scrum is no different from other approaches. The advantage of Scrum is that new requirements can easily be added into the release at sprint boundaries as long as low prority requirements scheduled later can be removed and still keep the project on time and on budget.

Application or Product Focus

For applications or products – either for the market or for internal use within an organization – Scrum moves groups away from the older project-centric model toward a continuous application/product development model. There is no longer a project with a beginning, middle, and end. And hence no traditional project manager. Rather, there is simply a stable Product Owner and a long-lived self- managing Team that collaborate in an "endless" series of two or four-week Sprints, until the product or application is retired. All necessary "project" management work is handled by the Team and the business owner—who is an internal business customer or from Product Management. It is not managed by an IT manager or someone from a Project Management Office.

Scrum can also be used for true projects that are one-time initiatives (rather than work to create or evolve long-lived applications); still, in this case the team and Product Owner do the project management.

What if there is insufficient new work from one or more existing applications to warrant a dedicated long-lived Team for each

application? In this case, a stable long-lived Team may take on items from one application in one Sprint, and then items from another in the next Sprint; in this situation the Sprints are often quite short, such as one week.

Occasionally, there is insufficient new work even for this last solution, and the Team may take on items from several applications during the same Sprint; however, beware this solution as it may devolve into unproductive multitasking across multiple applications. A basic productivity theme in Scrum is for the Team to be focused on one product or application for one Sprint.

Common Challenges

Scrum is not only a concrete set of practices – rather, and more importantly, it is a framework that provides visibility to the Team, and a mechanism that allows them to "inspect and adapt" accordingly. Scrum works by making visible the dysfunction and impediments that are impacting the Product Owner and the Team's effectiveness, so that they can be addressed. For example, the Product Owner may not really know the market, the features, or how to estimate their relative business value. Or the Team may be unskilled in effort estimation or development work.

The Scrum framework will quickly reveal these weaknesses. Scrum does not solve the problems of development; it makes them painfully visible, and provides a framework for people to explore ways to resolve problems in short cycles and with small improvement experiments.

Suppose the team fails to deliver what they committed to in the first Sprint due to poor task analysis and estimation skill. To the team, this feels like failure. But in reality, this experience is the necessary first step toward becoming more realistic and thoughtful about their commitments. This pattern – of Scrum helping make visible dysfunction, enabling the team to do something about it – is the basic mechanism that produces the most significant benefits that teams using Scrum experience.

Another common mistake is to assume that a practice is discouraged or prohibited just because Scrum does not specifically require it. For example, Scrum does not 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 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) are well advised.

Distributed, Outsourced Scrum

US, European, and Japanese companies often outsource software development to Eastern Europe, Russia, or the Far East. Typically, remote teams operate independently and communication problems limit productivity. While there is a large amount of published research on project management, distributed development, and outsourcing strategies as isolated domains, there are few detailed studies of best project management practices on large systems that are both distributed and outsourced.

Distributed Team Models

Here we consider three distributed Scrum models commonly observed in practice:

Isolated Scrums - Teams are isolated across geographies. In most cases off-shore teams are not cross-functional and may not be using the Scrum process.

Distributed Scrum of Scrums – Scrum teams are isolated across geographies and integrated by a Scrum of Scrums that meets regularly across geographies.

Totally Integrated Scrums – Scrum teams are cross- functional with members distributed across geographies. In the SirsiDynix case (see chapter 5), the Scrum of Scrums was localized with all Scrum Masters in Utah.

Most outsourced development efforts use a degenerative form of the Isolated Scrums model where outsourced teams are not cross-functional and not Agile. Requirements may be created in the U.S. and developed in Dubai, or development may occur in Germany and quality assurance in India. Typically, cross-cultural communication problems are compounded by differences in work style in the primary organization vs. the outsourced group. In the worst case, outsourced teams are not using Scrum and their productivity is typical of waterfall projects further delayed by cross-continent communications lag time.

Best practice recommended by the Scrum Alliance is a Distributed Scrum of Scrums model. This model partitions work across cross-functional, isolated Scrum teams while eliminating most dependencies between teams. Scrum teams are linked by a Scrum-of-Scrums where Scrum Masters (team leaders/project managers) meet regularly across

locations. This encourages communication, cooperation, and cross-fertilization and is appropriate for newcomers to Agile development.

An Integrated Scrums model has all teams fully distributed and each team has members at multiple locations. While this appears to create communication and coordination burdens, the daily Scrum meetings help to break down cultural barriers and disparities in work styles. On large enterprise implementations, it can organize the project into a single whole with an integrated global code base. Proper implementation of this approach provides location transparency and performance characteristics similar to small co-located teams.

Part Three

Scrum at Work

Chapter Four

Scrum Cases

This chapter serves as a retrospective on the origins of Scrum, its evolution in different companies, and a few key learnings along the way. It will provide a reference point for further investigation and implementation of Scrum.

Case 1: Easel Corporation

The First Scrum

Scrum was started in 1993 for software teams at Easel Corporation, where Jeff Sutherland was VP of object technology hired as chief engineer to lead a small team developing the first object-oriented design and analysis tool that incorporated round-trip engineering and automated object-relational mapping in enterprise development.

"There were some key factors that influenced the introduction of Scrum at Easel Corporation. The book Wicked Problems, Righteous Solutions by Peter DeGrace and Leslie Hulet Stahl reviewed the reasons why the waterfall approach to software development does not work for software development today. Requirements are not fully understood before the project begins. The users know what they want only after they see an initial version of the software. Requirements change during the software construction process. And new tools and technologies make implementation strategies unpredictable. DeGrace and Stahl reviewed "All-at-Once" models of software development, which uniquely fit object-oriented implementation of software and help to resolve these challenges.

All-at-Once model

All-at-Once models of software development assume that the creation of software is done by simultaneously working on requirements, analysis, design, coding, and testing and then delivering the entire system all at once. The simplest All-at-Once model is a single super-programmer creating and delivering an application from beginning to end. All aspects of the development process reside in a single person's head. This is the fastest way to deliver a product that has good internal architectural consistency, and it is the hacker's mode of implementation. The next level of approach to All-at-Once development is handcuffing two programmers together, as in the XP practice of pair programming.

Two developers deliver the entire system together. This is been shown to deliver better code (in terms of usability, maintainability, flexibility, and extendability) faster than work delivered by larger teams. The challenge is to achieve a similar overall productivity effect with an entire team and then with teams of teams.

Our team-based All-at-Once model was based on both the Japanese approach to new product development, Sashimi, and Scrum. We were already using production prototyping to build software. It was implemented in slices (Sashimi) where an entire piece of fully integrated functionality worked at the end of an iteration. What intrigued us was Hirotaka Takeuchi and Hujiro Nonaka's description of the team-building process in setting up and managing a Scrum. The idea of building a self-empowered team in which everyone had a global view of the product on a daily basis seemed like the right idea. This approach to managing the team, which had been so successful at

Honda, Canon, and Fujitsu, resonated with the systems thinking approach being promoted by Peter Senge at MIT.

We were also impacted by recent publications in computer science. As I alluded above, Peter Wagner at Brown University demonstrated that it was impossible to fully specify or test an interactive system, which is designed to respond to external inputs (Wegner's lemma). Here was mathematical proof that any process that assumed known inputs, as does the waterfall method, was doomed to failure when building an object-oriented system.

We were prodded into setting up the first Scrum meeting after reading James Coplien's paper on Borland's development of Quattro Pro for Windows. The Quattro team delivered one million lines of C++ code in 31 months, with a four person staff growing to eight people later in the project. This was about a thousand lines of deliverable code per person per week, probably the most productive project ever documented. The team attained this level of productivity by intensive interaction in daily meetings with project management, product management, developers, documenters, and quality assurance staff.

Software Evolution and Punctuated Equilibrium

Our daily meetings at Easel were disciplined in a way that we now understand as the Scrum pattern. The most interesting effect of Scrum on Easel's development environment was an observed "punctuated equilibrium effect." A fully integrated component design environment leads to rapid evolution of a software system with emergent, adaptive properties, resembling the process of punctuated equilibrium observed in biological species.

By having every member of the team see every day what every other team member was doing, we began to see how we could accelerate each other's work. For instance, one developer commented that if he changed a few lines in code, he could eliminate days of work for another developer. This effect was so dramatic that the project accelerated to the point where it had to be slowed down. This hyperproductive state was seen in several subsequent Scrums, but never went so dramatic as the one at Easel.

Achieving a Sustainable HyperProductive State

The key to entering a hyperproductive state was not just the Scrum organizational pattern. It was a combination of:

- The skill of the team
- The flexibility of a Smalltalk development environment
- The implementation of what are now know as XP engineering practices
- The way we systematically stimulated production prototypes that rapidly evolved into a deliverable product.

Furthermore, in the hyperproductive state, the initial Scrum entered what professional athletes and martial artists call "the zone." No matter what happened or what problems arose, the response of the team always was far better than the response of any individual. It was reminiscent of the Celtics basketball team at their peak, when they could do no wrong. The impact of entering the zone was not just hyperproductivity. People’s personal lives were changed. Team members said they would never forget working on the project, and they would always be looking for another experience like it. It induced open, team-oriented, fun-loving behavior in unexpected persons. Those individuals who could not function well in an open,

hyperproductive environment self-selected themselves out of the team by finding other jobs. This reinforced positive team behavior similar to biological systems, which select for fitness to the environment, resulting in improved performance of individual organisms."

Case 2: VMARK

The First Senior Management Scrum

When Easel Corporation was acquired by VMARK (subsequenctly Informix, Ascension Software, and now IBM) the original Scrum team continued its work on the same product. The VMARK senior management team was intrigued by Scrum and asked Jeff Sutherland to run a weekly senior management team Scrum to drive all the company's products to the Internet.

"These meetings started in 1995, and within a few months, the team had caused the introduction of two new Internet products and repositioned current products as Internet applications. Some members of this team left VMARK to become innovators in emerging Internet companies, so Scrum had an early impact on the Internet.
It was also at VMARK that Ken Schwaber was introduced to Scrum. Ken and I had worked together on and off for years. I showed him Scrum and he agreed it worked better than other project management approaches and was similar to how he built project management software in his company. He quickly sold off the project management software business and worked on bringing Scrum to the software industry at large. His work has had an incredible effect on deploying Scrum worldwide."

Case 3: Individual, Inc.

The First Internet Scrum

In the spring of 1996, Jeff Sutherland returned to Individual, Inc., a company he co-founded as VP of Engineering in 1988. Much of the Scrum experience at Individual has been documented by Ken Schwaber:

"The most impressive thing to me about Scrum at Individual was not that the team delivered two new Internet products – and multiple releases of one of the products – in a single quarter. It was the fact that Scrum eliminated several hours a day of senior management meeting time starting the day that Scrum began, within a week of my arrival at the company."

Because Individual had just gone public at the beginning of the Internet explosion, there were multiple competing priorities and constant revision of market strategy. As a result, the development team was constantly changing priorities and unable to deliver product. The management team was meeting daily to determine status of priorities that were viewed differently by every manager. These meetings were eliminated and the Scrum meetings became the focus for all decision making.

It was incredibly productive to force all decisions to occur in the daily Scrum meeting. If anyone wanted to know the status of specific project deliverables or wanted to influence any priority, he or she could only do it in the daily Scrum meeting. I remember the senior VP of marketing sat in on every meeting for a couple of weeks sharing her desperate concern about meeting Internet deliverables and timetables. The effect on the team was not to immediately respond to her despair. Over a period of two weeks, the team self- organized around a plan to

meet her priorities with achievable technical delivery dates. When she agreed to the plan, she no longer had to attend any Scrum meetings. The Scrum reported status on the Web with green lights, yellow lights, and red lights for all pieces of functionality. In this way, the entire company knew status in real time, all the time. This transparency of information has become a key characteristic of Scrum."

Case 4: IDX Systems

The First Scrum in the Large

During the summer of 1996, IDX Systems hired Jeff Sutherland as senior VP of engineering and product development. IDX had over 4,000 customers and was one of the largest US healthcare software companies, with hundreds of developers working on dozens of products. Here was an opportunity to extend Scrum to large-scale development.

"The approach at IDX was to turn the entire development group into an interlocking set of Scrums. Every part of the organization was team based including the management team, which included two vice presidents, a senior architect, and several directors. Front-line Scrums met daily. A Scrum of Scrums, which included the team leaders of each Scrum in a product line, met weekly, The management Scrum met monthly.

The key learning at IDX was that Scrum scales to any size. With dozens of teams in operation, the most difficult problem was ensuring the quality of the Scrum process in each team, particularly when the entire organization had to learn Scrum all at once. IDX was large enough to bring in productivity experts to monitor throughput on every

project. While most teams were only able to double the industry average in function points per month delivered, several teams moved into a hyperproductive state, producing deliverable functionality at four to five times the industry average. These teams became shining stars in the organization and examples for the rest of the organization to follow.

One of the most productive teams at IDX was the Web Framework team that built a web front-end infrastructure for all products. The infrastructure was designed to host all IDX applications, as well as seamlessly interoperate with end user or third party applications. It was a distributed team with developers in Boston, Seattle, and Vermont who met by teleconference in a daily Scrum meeting. The geographic transparency of this model produced the same high performance as co-located teams and has become the signature of hyperproductive distributed/outsourced Scrums."

The innovation and quality of this team's work continued to be demonstrated ten years later when IDX was acquired by GE Healthcare. The web framework was selected as the standard for GE applications.

Case 5: PatientKeeper

The First Scrum Company

In early 2000, Jeff Sutherland joined PatientKeeper, Inc. as chief technology officer and began introducing Scrum into a startup company. He was the 21st employee, and the development team grew from a dozen people to 45 people in six months. "PatientKeeper deploys mobile devices in healthcare institutions to capture and process financial and clinical data. Server technology synchronizes the

mobile devices and moves data to and from multiple back-end legacy systems. A robust technical architecture provides enterprise application integration to hospital and clinical systems. Data is forward-deployed from these systems in a PatientKeeper clinical repository. Server technologies migrate changes from our clinical repository to a cache and then to data storage on the mobile device. PatientKeeper proves that Scrum works equally well across technology implementations.

The key learning at PatientKeeper involved the introduction of eXtreme Programming techniques as a way to implement code delivered by a Scrum organization. While all teams seem to find it easy to implement a Scrum organizational process, they do not always find it easy to introduce XP. We were able to do some team programming and constant testing and refactoring, particularly as we migrated all development to Java and XML. It was more difficult to introduce these ideas when developers were working in C & C++. After a year of Scrum meetings in all areas of development, processes matured enough to capitalize on Scrum project management techniques, which were fully automated.

Complete automation and transparency of data allowed PatientKeeper to multithread Sprints through multiple teams. That in combination with implementing a MetaScrum of senior stakeholders in the company allowed PatientKeeper to run from top to bottom as a Scrum and become the first Scrum company to enter the hyperproductive state, delivering over 45 production releases a year of a large enterprise software platform. This became the prototype for the All-at-Once, or Type C Scrum, implemented in at least five companies by 2006.

PatientKeeper was the first company to achieve a hyperproductive revenue state driven by Scrum in 2007. Revenue quadrupled from 13M to 50M in one year.”

Other Prominent Projects

One of the most interesting things about Scrum is the unique case studies that have been published at IEEE conferences. Scrum is used by some of the most productive, high maturity, and most profitable software development teams in the world. Scrum powers:
The most productive large development project ever documented (see next chapter).

The most unique CMMI Level 5 implementation on the planet.
The most profitable software development project in the history of software development.

Systematic Software Engineering – a unique CMMI Level 5 implementation

Systematic Software Engineering in Aarhus, Denmark, spent seven years and over 100,000 person hours of process engineers to achieve CMMI Level 5 certification, reduce rework by 80%, and improve productivity by 31%. Within six months after a Scrum Certification course they had reduced planning time by 80%, defects by 40%, total cost of a project by 50% while simultaneously enhancing customer and employee satisfaction. They now bid Scrum projects at 50% of the cost of waterfall projects.

Google AdWords – the most profitable development project in history

One of the most interesting Scrum projects is Google's AdWords implementations. This application drives the majority of Google revenue growth and helps create market capitalization that is higher than Intel and just below that of Chevron, the most profitable oil company in the world. The AdWords project, powered by Scrum, has distributed teams in five locations and interfaces with virtually all Google products on every release. As a result, the Google project manager needed to insert more structure than is usually associated with Google teams. His seamless introduction of Scrum based on resolving the highest priority impediments observed by the teams resulted in an

Results From a Scrum Project at Yahoo!

How does Scrum work, compared to the approach which was used previously at the company?

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 on average by 37%, based on the estimates of the Product Owners.

Buy In: 86% of team-members stated that they would continue using Scrum if the decision were solely up to them.

(Based on a quarterly survey 2007, including everyone at Yahoo! using Scrum, i.e. Product Owners, Team Members, Scrum Masters, and the functional managers of those individuals.)

implementation that no longer needed a Scrum Master to function. The teams ran by themselves.

Chapter Five

The SirsiDynix case

This case study summarizes an extraordinary distributed project that ran smoothly across ten time zones.

It may seem improbable, but during the most productive Java project ever documented, the 56 developers from SirsiDynix and StarSoft Development Laboratories had an ocean and half a continent between them. Working from Provo in Utah, Waterloo in Canada and St. Petersburg in Russia, the distributed team delivered 671,688 lines of production Java code during 2005. In total, the Java application consisted of over 1,000,000 lines of code. This proves that a large, distributed, outsourced team actually can achieve a hyperproductive state – in this case 15.3 function points per developer & month.
Best practices for distributed Scrum seen on this project consisted of: daily Scrum team meetings of all developers from multiple sites daily meetings of the Product Owner team hourly automated builds from one central repository no distinction between developers at different sites on the same team seamless integration of XP practices like pair programming with Scrum

The Companies

SirsiDynix has approximately 4,000 library and consortia clients, serving over 200 million people through over 20,000 library outlets in the Americas, Europe, Africa, the Middle East and Asia-Pacific. Jack Blount, President and CEO of Dynix and now CTO of the merged SirsiDynix company, negotiated an outsource agreement with StarSoft who staffed the project with over 20 qualified engineers in 60 days. Significant development milestones were completed in a few weeks and joint development projects are efficiently tracked and continue to be on schedule.

StarSoft Development Labs, Inc. is a software outsourcing service provider in Russia and Eastern Europe. Headquartered in Cambridge, Massachusetts, USA, StarSoft operates development centers in St. Petersburg, Russia and Dnepropetrovsk, Ukraine, employing over 450 professionals. StarSoft has experience handling development efforts varying in size and duration from just several engineers working for a few months to large-scale projects involving dozens of developers and spanning several years. StarSoft successfully uses Agile development and particularly XP engineering practices to maintain CMMI Level 3 certification and was acquired by Exigen Services in 2007.

A Huge Task at Hand

SirsiDynix was confronted with the requirement to completely re-implement a legacy library system with over 12,500 installed sites. Large teams working over many years in a changing business environment faced many new requirements in the middle of the project. To complicate matters further, the library software industry was in a consolidating phase. Dynix started the project in 2002 and merged with Sirsi in 2005 to form SirsiDynix.

Fortunately, Dynix started with a scalable Agile process that could adapt to changing requirements throughout the project. Time to market demanded more than doubling of output. That could only happen by augmenting resources with Agile teams. StarSoft was selected because of their history of successful XP implementations and their experience with systems level software.

The combination of high risk, large scale, changing market requirements, merger and acquisition business factors, and the SirsiDynix experience with Scrum combined with StarSoft success with XP led them to choose an Integrated Scrums implementation. Jack Blount's past experience with Agile development projects at US Data Authority, TeleComputing and JD Edwards where he had used Isolated Scrums and Distributed Scrum of Scrums models did not meet his expectations. This was a key factor in his decision to structure the project as Integrated Scrums.

The Systems and Software Consortium (SSCI) has outlined drivers, constraints, and enablers that force organizations to invest in real-time project management information systems. Scalable Scrum implementations with minimal tooling are one of the best real-time information generators in the software industry.

SSCI complexity drivers are described as:

- Increasing **problem complexity** shifting focus from requirements to objective capabilities that must be met by larger teams and strategic partnerships.
- Increasing **solution complexity**, which shifts attention from platform architectures to enterprise architectures and fully integrated systems.

- Increasing **technical complexity** from integrating standalone systems to integrating across layers and stacks of communications and network architectures.
- Increasing **compliance complexity** shifting from proprietary to open standards.
- Increasing **team complexity** shifting from a single implementer to strategic teaming and mergers and acquisitions.

SirsiDynix faced all of these issues. Legacy products were difficult to sell to new customers. They needed a new product with complete functionality for the library enterprise based on new technologies that were highly scalable, easily expandable, and used the latest computer and library standards,

Top Issues in Distributed Development

The SSCI has carefully researched top issues in distributed development, all of which had to be handled by SirsiDynix and StarSoft.

- **Strategic:** Difficulty leveraging available resources, best practices are often deemed proprietary, are time consuming and difficult to maintain.
- **Project and process management**: Difficulty synchronizing work between distributed sites.
- **Communication:** Lack of effective communication mechanisms.
- **Cultural:** Conflicting behaviors, processes, and technologies. Technical: Incompatible data formats, schemas, and standards.
- **Security:** Ensuring electronic transmission confidentiality and privacy.

The unique way in which SirsiDynix and StarSoft implemented an Integrated Scrums model carefully addressed all of these issues.

Solution: Integrated Scrums

SirsiDynix used the three scrum roles – Scrum Master, Product Owner & Team – to solve the strategic distribution problem of building a high velocity, real-time reporting organization with an open source process that is easy to implement and low-overhead to maintain.
For large programs, a Chief ScrumMaster to run a Scrum of Scrums and a Chief Product Owner to centrally manage a single consolidated and prioritized product backlog is essential. SirsiDynix located the Scrum of Scrums and the Product Owner teams in Utah.

Team Formation

The second major challenge for large projects is process management, particularly synchronizing work between sites. This was achieved by splitting teams across sites and fine tuning daily Scrum meetings. Teams at SirsiDynix were split across the functional areas needed for an integrated library system. Half of a Scrum team is typically in Provo, Utah, and the other half in St. Petersburg. There are usually 3-5 people on the Utah part of the team and 4 or more on the St. Petersburg portion of the team. The Search and Reporting Teams are smaller. There are smaller numbers of team members in Seattle, Denver, St. Louis, and Waterloo, Canada.

Scrum Meetings

Teams meet across geographies at 7:45am Utah time which is 17:45 St. Petersburg time. Teams found it necessary to distribute answers to

the three Scrum questions in writing before the Scrum meeting. This shortens the time needed for the join meeting teleconference and helps overcome any language barriers. Each individual reports on what they did since the last meeting, what they intend to do next, and what impediments are blocking their progress.

Email exchange on the three questions before the daily Scrum teleconference was used throughout the project to enable phone meetings to proceed more smoothly and efficiently. These daily team calls helped the people in Russia and the U.S. learn to understand each other. In contrast, most outsourced development projects do not hold formal daily calls and the communication bridge is never formed. Local sub-teams have an additional standup meeting at the beginning of the day in St. Petersburg. Everyone uses the same process and technologies and daily meetings coordinate activities within the teams.

Scrum Masters are all in Provo, Utah or Waterloo, Canada, and met in a Scrum of Scrums every Monday morning. Here work is coordinated across teams. Architects are directly allocated to production Scrum teams and all located in Utah. An Architecture group also meets on Monday after the Scrum of Scrums meeting and controls the direction of the project architecture through the Scrum meetings. A Product Owner resident in Utah is assigned to each Scrum team. A chief Product Owner meets regularly with all Product Owners to assure coordination of requirements.

SirsiDynix achieved strong central control of teams across geographies by centrally locating Scrum Masters, Product Owners, and Architects. This helped them get consistent performance across distributed teams.

Sprints

Sprints are two weeks long on the SirsiDynix project. There is a Sprint planning meeting similar to an XP release planning meeting in which requirements from User Stories are broken down into development tasks. Most tasks require a lot of questions from the Product Owners and some tasks take more time than initial estimates.

The lag time for Utah Product Owner response to questions on User Stories forces multitasking in St. Petersburg and this is not an ideal situation. Sometimes new tasks are discovered after querying Product Owners during the Sprint about feature details.

Code is feature complete and demoed at the end of each Sprint.
Up until 2006, if it met the Product Owner's functional requirement, it was considered done, although full testing was not completed. It was not deliverable code until SirsiDynix strengthened its definition of "done" to include all testing in 2006. Allowing work in progress to cross Sprint boundaries introduces wait times and greater risk into the project. It violates the lean principle of reducing work in progress and increases rework.

Product Specifications

Requirements are in the form of User Stories used in many Scrum and XP implementations. Some of them are lengthy and detailed, others are not. A lot of questions result after receiving the document in St. Petersburg which are resolved by daily Scrum meetings, instant messaging, or email.

For this project, St. Petersburg staff like a detailed description because the system is a comprehensive and complex system designed for

specialized librarians. As a result, there is a lot of knowledge that needs to be embedded in the product specification.

The ways libraries work in St. Petersburg are very different than English libraries. Russian libraries operate largely via manual operations. While processes look similar to English libraries on the surface, the underlying details are quite different. Therefore, user stories do not have sufficient detail for Russian programmers.

Story for Simple Renewals Use Case:

Patron brings book or other item to staff to be renewed. Patron John Smith checked out "The Da Vinci Code" the last time he was in the library. Today he is back in the library to pick up something else and brings "The Da Vinci Code" with him. He hands it to the staff user and asks for it to be renewed. The staff user simply scans the item barcode at checkout, and the system treats it as a renewal since the item is already checked out to John. This changes the loan period (extends the due date) for the length of the renewal loan. Item and patron circulation history are updated with a new row showing the renewal date and new due date. Counts display for the number of renewals used and remaining. The item is returned to Patron John Smith.

Assumptions:
Item being renewed is currently checked out to the active patron

- No requests or reservations outstanding
- Item was not overdue
- Item does not have a problem status (lost, etc)
- No renew maximums have been reached
- No block/circulation maximums have been reached
- Patron's subscriptions are active and not within renewal period
- No renewal charges apply
- No recalls apply

Renewal is from Check Out (not Check In) Staff User has renewal privileges Verification (How to verify completion):

- Launch Check Out
- Retrieve a patron who has an item already checked out but not yet overdue
- Enter barcode for checked out item into barcode entry area (as if it is being checked out), and press <cr>.
- System calculates new due date according to circulation rules and agency parameters.
- The renewal count is incremented (Staff renewal with item)
- If user views "Circulation Item Details", the appropriate Renewals information should be updated (renewals used/remaining)
- Cursor focus returns to barcode entry area, ready to receive next scan (if previous barcode is still displayed, it should be automatically replaced by whatever is entered next)
- A check of the item and patron circulation statistics screens shows a new row for the renewal with the renewal date/time and the new due date.

Testing

Developers write unit tests. The Test team and Product Owners do manual testing. An Automation Test team in Utah creates scripts for an automated testing tool. Stress testing is as needed.

During the Sprint, the Product Owner tests features that are in the Sprint backlog. Up until 2006, testers received a stable Sprint build only after the Sprint demo. The reason for this was a lower tester/ developer ratio than recommended by the Scrum Alliance.

There are 30 team members in North America and 26 team members in St. Petersburg on this project. The St. Petersburg team has one project leader, 3 technical team leaders, 18 developers, 1 test lead, and 3 testers. This low tester/developer ratio initially made it impossible to have a fully tested package of code at the end of the Sprints.

The test-first approach was initially encouraged and not mandated. Tests were written simultaneously with code most of the time. GUIs were not unit tested.

Functional Test Example:

Functional Area	Reserve Book Room
Task Description	Check that items from Item List is placed under Reserve with "Inactive" status
Condition	1. User has right to place Items under Reserve 2. At least one Item List exists in the system 3. Default Reserve Item Status in Session Defaults is set to "Inactive"

Entry Point	Launcher is opened
Test Data	No specific data
Action	1. Reserve > Reserve Item 2. Select "Item Search" icon 3. Select "Item List" in the Combo box list of search options and enter appropriate Item list name 4. Press Enter 5. Select all Items which appear in the Item Search combo box and press "OK"
Expected Results	1. Items that were in Item list should appear in the list in Reserve Item 2. Status of all items that has been just added should be shown as "Inactive" 3. Save button should be inactive 4. All corresponding Items should retain their original parameters

In the summer of 2006, a new CTO of SirsiDynix, Talin Bingham, took over the project and introduced Test Driven Design. Every Sprint starts with the usual Sprint Planning meeting and teams are responsible for writing functional tests before doing any coding. Once functional tests are written and reviewed, coding starts. Test-first coding is mandated.

When coding is complete, developers run unit tests and manually pass all the functional tests before checking in changes to the repository. Automation testing is done using the Compuware TestPartner tool, but there is still room for improvement of test coverage.

Configuration Management

SirsiDynix was using CVS as source code repository when the decision was made to engage an outsourcing firm. At that time, SirsiDynix made a decision that CVS could not be used effectively because of lack of support for distributed development, largely seen in long code

synchronization times. Other tools were evaluated and Perforce was chosen as the best solution.

StarSoft had seen positive results on many projects using Perforce. It is fast, reliable and offers local proxy servers for distributed teams. Although not a cheap solution, it has been very effective for the SirsiDynix project.

Automated builds run every hour with email generated back to developers. It takes 12 minutes to do a build, 30 minutes if the database changes. StarSoft would like to see faster builds and true concurrent engineering. Right now builds are only stable every two weeks at Sprint boundaries.

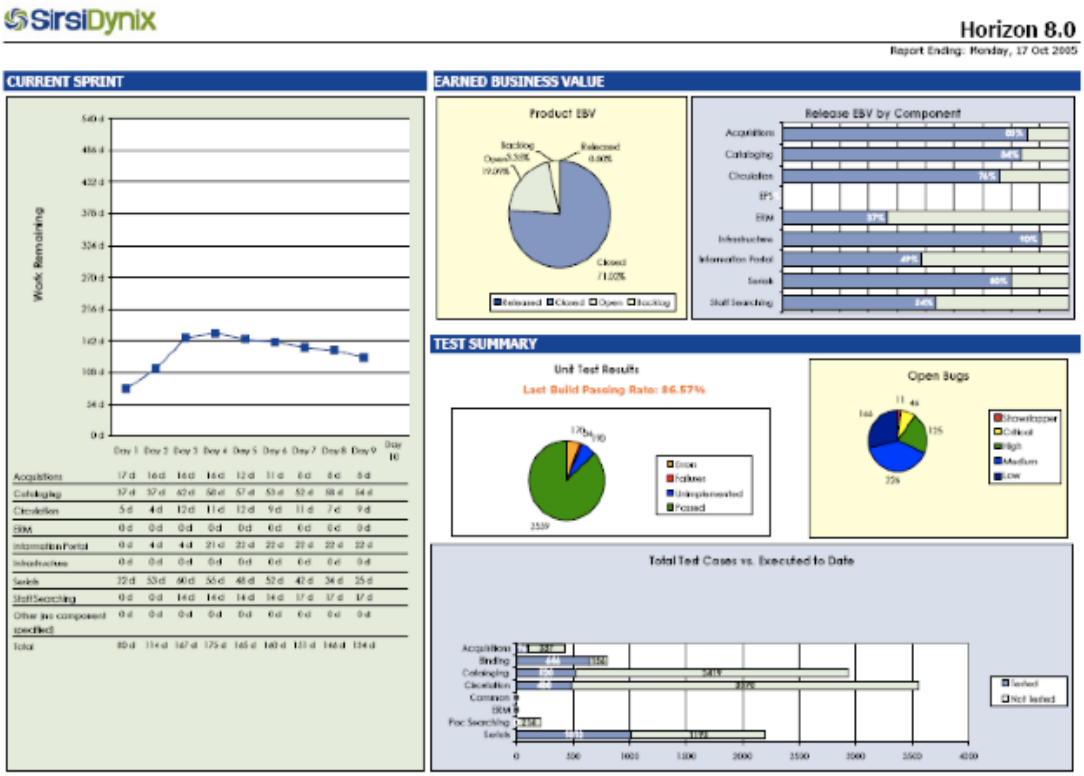
Pair Programming, Refactoring, and Other XP Practices

StarSoft is an XP company and tries to introduce XP practices into all their projects. Pair programming is done on more complicated pieces of functionality. Refactoring was planned for future Sprints and not done in every iteration as in XP. Some radical refactoring without loss of functionality occurred as the project approached completion. Continuous integration is implemented as hourly builds.

On this project, these three engineering practices were used with Scrum as the primary project management methodology.

Measuring Progress

The project uses the Jira issue tracking and project management software. This gives everyone on the project a real-time view into the state of Sprints. It also provides comprehensive management reporting tools.



SirsiDynix Horizon 8.0 Project Dashboard showing the Sprint burn-down chart, a snapshot of Earned Business, and a synopsis of bug status

Data from Jira can be downloaded into Excel to create any requested data analysis. High velocity projects need an automated tool to track status across teams and geographies. The best tools support bug tracking and status of development tasks in one system and avoid extra work on data entry by developers. Such tools should track tasks completed by developers and work remaining. They provide more detailed and useful data than time sheets, which should be avoided. Time sheets are extra overhead that do not provide useful information on the state of the project, and are de-motivating to developers.

Resulting Context with Integrated Scrums

Collaboration of SirsiDynix and StarSoft turned the Horizon 8.0 project into one of the most productive Scrum projects ever documented. For example, data is provide in the table below on a project that was done initially with a waterfall team and then re- implemented with a Scrum team. The waterfall team took 9 months with 60 people and generated 54000 lines of code. It was re- implemented by a Scrum team of 4.5 people in 12 months. The resulting 50,803 lines of code had more functionality and higher quality.

	Scrum	Waterfall	SirsiDynix
Person Months	54	540	827
Java LOC	50.083	54000	671.688
Function Points	959	900	12673
FP per dev/ month	17.8	2.0	15.3

Function Points/Developer Month for collocated vs. distributed projects.

Capers Jones of Software Productivity Research has published extensive tables on average number of function points per lines of code for all major languages. Since the average lines of code per function point for Java is 53, we can estimate the number of function points in the Scrum application. The waterfall implementation is known to have fewer function points.

Distributed teams working on Horizon 8.0 generated 671,688 lines of code in 14.5 months with 56 people. During this period they radically refactored the code on two occasions and reduced the code base by 275,000. They have not been penalized for refactoring as that is rarely done in large waterfall projects in the database from

which Capers derived his numbers. They have also not been rewarded for refactoring even though reducing lines of code is viewed as important as adding new code on well-run Agile projects.

Jones has also shown from his database of tens of thousands of projects that industry average productivity is 12.5 function points per developer/month for a project of 900 function points and that this drops to 3 for a project with 13000 function points. Some of this is due to 4GL and other code-automation tools used on small projects, many of which are not implemented in third generation languages like Java.

The SirsiDynix project is almost as productive as the small Scrum project with a collocated team of 4.5 people. For a globally dispersed team, it is one of the most productive projects ever documented at a run rate of five times industry average.

CHAPTER 6

Can Scrum projects fail?

Let's face it: although the success rate is astonishingly high, scrum projects sometimes fail – most often due to poor leadership.

Scrum can be seen as a framework for continuous process improvement. Harvard Professor John Kotter notes that 70% of change processes fail, primarily due to a lack of sense of urgency among the leadership.

Scrum is very resilient with a success rate of over 70% according to the latest worldwide survey of over 2000 companies (Version One survey). Yet it is not a silver bullet and leadership failure is the primary cause of Scrum failure. Let's look at two examples.

Case Study 1: "EmbeddedWaterFall.com"

- Roman Pichler of Pichler Consulting Ltd., London

At a development organization specializing in embedded communications products, the head of development was determined to implement Scrum in order to get faster delivery, higher quality from software teams. Let's call this company EmbeddedWaterFall.com. A pilot project to create a new software system was selected. Success would strongly influence the future of the development organization.

Product management at EmbeddedWaterFall.com was skeptical about Agile processes. Three Scrum teams created an architectural baseline/internal release in six months of two-week iterations which included deployment. Developers embraced Agile practices well, the head of development was pleased with results, and product management liked the transparency of reporting. The Human Resource department was encouraged to align performance appraisals with agile practice as it is "fun", a "challenge", and a "positive change."

Then impediments begin to appear. Scaling to eight teams is difficult. Velocity of software delivery is lower than expected. Re- organization at headquarters leads to loss of support for Agile practice from product management. The head of development now insists that date and scope must be met. Command and control with task-based planning is implemented along with overtime and weekend work. Most agile development practices are abandoned. EmbeddedWaterFall.com reverted to type.

There was an extensive analysis of root causes and lessons learned on this project. The bottom line is failure of management to understand agile practice and failure of management commitment to implement Scrum made it impossible to remove impediments at the first sign of trouble.

Case Study 2: "GameOver.now"

by Henrik Kniberg of Crisp SA, Stockholm

A second case study shows how aggressive action can resolve management challenges when management is willing to adapt

and remove impediments to Scrum implementation. Let's call this company GameOver.now where Scrum was implemented for the most important project in the company to deliver a critical software application on a fixed date in April of 2007.

A Scrum team ran two-week sprints from April to September in 2006 to produce detailed requirements. Then they ran two-week sprints to code the requirements from October to December. January through March of 2007 was reserved for testing.

In January the code is not complete, testing has not begun, and the management is hovering over the team worried about progress. They call in an expert Scrum trainer who notices the team is not really a team. The DBA works independently on her set of tasks. A three person subgroup in the team mistrusts everyone else. Management is starting to micromanage an impending disaster. Waterfall has been implemented under a Scrum banner.

The Scrum trainer says it is time to implement Scrum. We will create a product backlog, estimate the product backlog, find the actual velocity of the team by running two sprints, and determine the release date by building a roadmap. Created and estimating stories using Planning Poker for incomplete product backlog showed that 180 points were remaining. There were 70 points of testing remaining for the portion of the backlog that had been coded. The team completed two sprints with a velocity of 10. At currentcapacity, the project would take 25 two-week sprints and be delivered a year late.

In order to improve the date, the size of the backlog needs to be

reduced and the velocity needs to be increased. However, the root cause of the current problem is management lack of focus. A companywide meeting is held and the top priority project is clearly explained to the entire staff. They are told not to disrupt the team, to help them whenever they can, and that this project is really the top priority for the company.

The team systematically removes impediments and triples their velocity to 30 points per sprint. They deliver an early release to the customer in the same quarter as the original schedule. The customer is both surprised and happy. They deliver a final incremental release during the next quarter. While the project was several months late, it was six months earlier than the waterfall Scrum would have delivered it.

The lesson here is that even a failed project can be rescued at the eleventh hour by Scrum if management and the team will actually implement Scrum.

Appendix 1

Who's Who in Scrum?

This is a list of people, organisations and teams who have inspired, informed and instructed Jeff Sutherland and his colleagues. Hence, they have all contributed to the creation of Scrum in their own special manner.

Hirotaka Takeuchi and **Ikujiro Nonaka**, the Godfathers of Scrum, unknowingly gave Scrum its name and helped create a global transformation of software development. " The New New Product Development Game". Harvard Business Review, (January-February 1986)

Jim Coplien and the **ATT Bell Labs Pasteur Project** wrote the paper on the most productive software development team ever documented the Borland Quattro Pro Project. The first Scrum team implemented the Scrum daily meeting after reading this paper.

Alan Kay and his team at Xerox Parc invented Smalltalk, the mouse, the graphical user interface, the personal computer, the Ethernet, and the laser printer. Listening to his insights on innovation inspired the first Scrum team to go from "good" to "great".

Professor **Rodney Brooks** launched the startup now known as iRobot in space leased from Jeff Sutherland. He taught the subsumption architecture, how to create simple rules to produce highly intelligent performance from complex adaptive systems.

Christopher Langton of Los Alamos Labs and the Sante Fe Institute coined the term "artificial life" and showing that increasing degrees of

freedom up to the edge of chaotic behavior accelerated their evolution. Scrum feels "chaotic" by intent, so as to accelerate software evolution.

The French object-database developers working near the MIT campus at Graphael (later Object Databases, then Matisse Software) were the first to demonstrate in Lisp and then in C++ the Agile patterns of pair programming, radical refactoring, continuous integration, common ownership of code, world class user interface design, and other tips and tricks which **Kent Bent** used to create eXtreme Programming a decade later. These were all incorporated into the first Scrum.

The **Creative Initiative Foundation** worked with Silicon Valley volunteers to help make the world a better place, the underlying motivation driving the founders of Scrum. This connected the Co-Creators of Scrum with the early systems thinking of MIT Professor **Peter Senge** who later wrote "The Fifth Discipline."

Capers Jones and his productivity experts at Software Productivity Research analyzed and reanalyzed the output of early Scrum teams, as well as many of the software products built with Scrum during 1994-2000. These analyses allowed the first Scrum team to provide a money-back guarantee that users would double productivity during the first month using tools created by the first Scrum.

The first Scrum team – **John Scumniotales** (ScrumMaster), Don Roedner (Product Owner), **Jeff McKenna** (Senior Consultant), **Joe Kinsella** (object-relational mapping), **Laurel Ginder** (QA), and three Danish developers - **Grzegorz Ciepiel**, **Bent Illum**, and **John Lindgreen**. They endured repeated failure, depressing analysis of these failures in front of their technical peers from other companies, and transcendence of their missteps. They were the first Scrum team

to achieve the hyperproductive state for which Scrum was designed and their product, Object Studio, was reported as industry leader by computer trade journals. Little did they know that Scrum would be their greatest contribution.

PatientKeeper Inc., the first company to fully implement an "All at Once" or Type C Scrum involving the entire company in Scrum practice. This innovation in process design was documented by **Mary and Tom Poppendieck** in their book *Lean Software Development*. "I find that the vast majority of organizations are still trying to do too much stuff, and thus find themselves thrashing. The only organization I know of which has really solved this is Patient Keeper."

Christopher Alexander coined the phrase " quality without a name" (QWAN) – something that many Scrum practitioners experience. The phrase was used in the book "The Timeless Way of Building", where Alexander describes a certain quality that we seek, but which cannot be named. This may be the most important feature of Scrum and can only be spoken of as a set of core values - openness, focus, commitment, courage, and respect. It could be viewed as the "speed of trust" or one of the sources of "ba" often seen on Scrum teams. Ba is the Japanese term for the creative flow of innovation described by Takeuchi and Nonaka.

Appendix 2

References

For a complete list of Jeff Sutherland papers, please visit http://scrum.jeffsutherland.com/

C. Jakobsen and J. Sutherland, "Scrum and CMMI – Going from Good to Great: are you ready-ready to be done-done?," in Agile 2009, Chicago, 2009.

K. Schwaber and J. Sutherland. The Scrum Guide. Scrum.org, 2010.

A. Sutherland, J. Sutherland, and C. Hegarty, "Scrum in Church: Saving the World One Team at a Time," in Agile 2009, Chicago, 2009.

J. Sutherland, "Future of Scrum: Parallel Pipelining of Sprints in Complex Projects," in AGILE 2005 Conference Denver, CO: IEEE, 2005.

J. Sutherland and I. Altman, "Organizational Transformation with Scrum: How a Venture Capital Group Gets Twice as Much Done with Half the Work," in 43rd Hawaii International Conference on Software Systems, Kauai, Hawaii, 2010.

J. Sutherland, S. Downey, and B. Granvik, "Shock Therapy: A Bootstrap for a Hyper- Productive Scrum" in Agile 2009, Chicago, 2009.

J. Sutherland, G. Schoonheim, and M. Rijk, "Fully Distributed Scrum: The Secret Sauce for Hyperproductive Offshored Development Teams," in Agile 2008, Toronto, 2008.

J. Sutherland and K. Schwaber, The Scrum Papers: Nuts, Bolts, and Origins of an Agile Method. Boston: Scrum, Inc., 2007.

J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov, "Distributed Scrum: Agile Project Management with Outsourced Development Teams," in HICSS'40, Hawaii International Conference on Software Systems Big Island, Hawaii: IEEE, 2007.

Takeuchi and I. Nonaka, “The New New Product Development Game,” Harvard Business Review, 1986.