An Introduction to Scrum

Giovanni Asproni, gasproni@asprotunity.com

Introduction

Scrum is one of the most well known agile methodologies. It has been invented by Ken Schwaber and Jeff Sutherland around 1993 (date of the first Scrum as reported in "Agile Development: Lessons Learned from the First Scrum" by Jeff Sutherland, which can be found in the resources section of the ScrumAlliance web-site [11]).

It has three main characteristics that make it very attractive. The first one is simplicity—its basics can be learnt in less than a day. The second one is that it provides value to all the stakeholders of the project—the customer, the users, the project manager, and also the developers. The third one is scalability.

It provides value to the customers—the people or organizations funding the project—and to the users. In fact its iterative and incremental nature, along with the prioritization of requirements, allows them to have the most valuable features available early—the most important ones, by the end of the first iteration. Furthermore, Scrum gives them total control on the direction and scope of the project, since they can decide at the end of each iteration to continue the project by adding new functionality or modifying existing one; or can terminate the project because the product is good enough, or has no value anymore.

It provides value to the project manager by providing tools—burn-down chart, product backlog, sprint backlog—and techniques—daily meetings, sprint planning meeting, sprint review meeting—aimed at increasing the visibility of all the aspects of the project. This visibility allows more control on the project, and earlier discovery of the problems that may happen during the project lifetime.

It provides value to the developers. They are usually highly motivated by interesting learning opportunities and technical challenges, so belonging to a cross-functional self-organizing team—such as a scrum team—is an ideal situation for them since they can get involved in all development activities—from discussing requirements to writing code.

Finally, even if Scrum works better with small teams—seven plus or minus two developers [15]—it scales up well with project size. In fact, it has been used in projects with up to several hundred developers [14].

However, its implementation can sometimes be difficult—Scrum, like all the other agile methodologies, is heavily based on teamwork, communication, trust, and on delegating responsibility and authority. All this things together represent a major cultural change—especially for companies used to more traditional methods—which, usually, requires time and hard work to be fully accepted. In this article—after introducing the basics of the methodology—I'll discuss some of those problems—along with some suggestions for their solution.

Before introducing Scrum I'll introduce the Agile Manifesto—the set of values and principles on which this methodology is based—along with a definition of what is Agile Software Development. This is the subject of the next section.

What is Agile Software Development: the Agile Manifesto

The term *Agile Software Development* is a container that includes all the methodologies that share the values and principles stated on the *Agile Manifesto* (see sidebar). The main characteristic of all these methodologies is that they are *people-centric*—they emphasise teamwork, face-to-face communication, and short feedback-loops—in contrast with more traditional ones, which, instead, are *process-centric*—they focus more on the sequence of activities to be executed and the tools used [2].

The most important goal that agile methodologies try to achieve is to deliver value to *all* the stakeholders, including the *developers*.

The value delivered to the customers may be obvious: software that is fit for its intended purpose on time and within budget.

The value delivered to the developers is much less obvious: job satisfaction and self-motivation. In fact, the usage of an agile methodology can be a big help in increasing and keeping developers' motivation [2] which is the most important factor influencing developers' productivity [6], which, in turn, has a direct positive impact on the value delivered to the customers.

As we'll discover in the rest of the article, Scrum obeys to all the values and principles of the Agile Manifesto (this should not be surprising since its inventors are both among its signatories).

Agile Manifesto

The Agile Manifesto has been taken from [5]

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

**That is, while there is value in the items on
the right, we value the items on the left more.**

Principles behind the Agile Manifesto

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective methodology of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity—the art of maximising the amount of work not done—is essential.

The best architectures, requirements, and designs emerge from self-organising teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Scrum Fundamentals

Scrum, like all the other agile methodologies, is *iterative*—the product is developed in a sequence of self contained mini-projects called iterations—and *incremental*—the product functionality grows incrementally by adding new features during each iteration. It comes also with its own terminology both for some of the roles of the people involved, and for some of the process activities. I'll introduce them in the next two sections. A quick overview of the process is given in **Error! No bookmark name given.**.

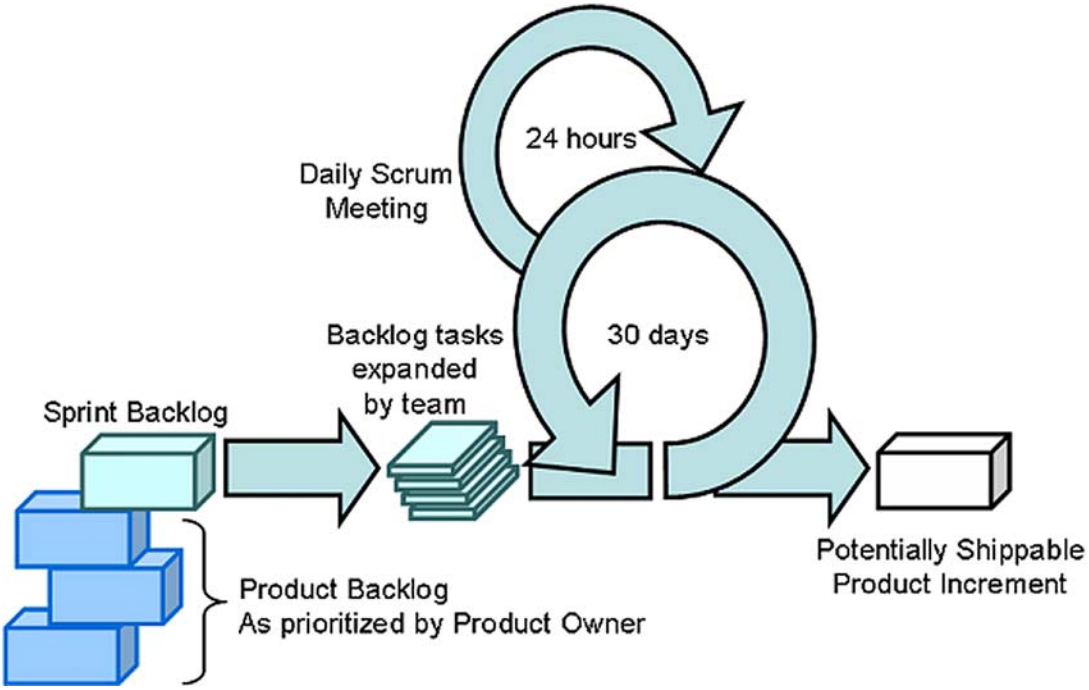

Figure 1: Scrum process overview adapted from [14]

Roles

In Scrum there are only three roles: *product owner*, *scrum master*, and *team member*.

The product owner is the responsible for deciding the functionality of the product, and the prioritization for its implementation. He represents the interests of everyone with a stake in the project and its final product—customers, users, etc. Often this role is covered by someone from the marketing team, or a key user. This role is the analogous of the *customer* role in the Agile Manifesto.

The scrum master is responsible for establishing scrum practices and rules, representing management to the project, shielding the team and removing obstacles—making sure, among the other things, that each team-member has proper hardware and software resources, desk space, etc. Often this role is covered by a technical team leader, or the project manager, and, sometimes, by the product owner.

A team member is someone belonging to the team that carries out the actual development activities. The team is cross functional—for a web-application it may include web-designers, developers, testers, database administrators, system administrators, technical writers, etc.—and self-organizing—its members decide how to proceed to a successful delivery without any external interference.

Team members, ideally, have no titles attached (such as architect, technical leader, etc.), instead they collaborate as peers, and each of them can be involved in any activity—design, development, documentation, etc.—independently of their area of expertise.

Team membership should be a full-time engagement. However, this is not always possible—especially for some roles such as database and system administrators which are typically shared by several teams in an organization. My preferred approach is to have a team of full-time members, and, if necessary, some part-time specialists—external to the team—available for giving expert help.

It is always advisable for anybody involved in a scrum project, to avoid covering more than one role at the same time. The reason is that each role comes with a different set of responsibilities that, sometimes, may clash with the ones of another role—for example, a product owner may try to push for the development team to work around the clock to make an impossible deadline, in which case a scrum master should take the part of the team and defend it against this abuse. However, this separation is not always possible. In that case, dealing with the potential conflicts is left to the common sense of the people involved.

Process Activities and Tools

In this section I'll introduce the vocabulary for the process related parts of Scrum.

The first important term is *sprint*, which is just another word for iteration. The suggested duration for a sprint is thirty calendar days, however it is common to find teams working with one or two-week long iterations. Once a duration is chosen it is important to keep it the same for each sprint in order to allow the team to find a proper rhythm, and to simplify the management and tracking activities.

At the beginning of each Sprint, there is a *sprint planning meeting*, which is a one-day long activity split in two parts.

The first one is a four hour planning session during which the product owner creates the *sprint backlog* by selecting as many high priority items as the team can commit to deliver in a sprint from the *product backlog*, which is the set of all the known requirements of the product. The product backlog may (and usually does) change over time, as new requirements are added, or existing ones are dropped.

After the creation of the sprint backlog, the product owner and the team define the *sprint goal*, which is the business value that the product increment must deliver regardless of functionality implemented. Its purpose is to give the team focus, and the ability to choose among different options in case of problems or uncertainty. A good sprint goal is measurable; in this way it is easier to know when it has been achieved. For example the goal for a web-application for a particular sprint might be "Handle two times more connections than version 2.0."

In the second part the team splits the sprint backlog into a set of *tasks*—units of work estimated between four and sixteen hours—to be executed to deliver the required functionality. This set is not meant to be complete, as some new tasks may emerge after the sprint has already started. The product owner doesn't participate to this meeting, but it has to be available for answering questions the team may have.

The sprint planning meeting is time-boxed to eight hours, and the first part is time-boxed to four hours.

At this point, is important to note that in Scrum time-boxing is very strict. When the time is expired, the time-boxed activity has to stop—for hours is exactly two-hundred and forty minutes, not a minute more. The reason for that is to let people concentrate on what is important and avoid wasting time in secondary issues.

Every day, at the same time, same place, and, ideally, first thing before starting to work, the team has a stand-up meeting (the *daily scrum*) lasting, at most, fifteen minutes—independently of the size of the team—in which each member answers three questions:

- What did you do yesterday?

- What will you do today?
- What obstacles are in your way?

The meeting is for synchronization purposes only (not for problems solving). Any issues are dealt with after the meeting is finished.

Attendance to the meeting is open to everybody with a interest in the project. The participants are split in two groups. The first one—the pigs— coincides with the development team; everybody else belongs to the second one —the chickens [14]. During the meeting only the pigs can talk, while the chickens can only observe silently. This rule helps to keep the meeting short and focused, and, at the same time, to keep every stakeholder informed about the progress made and the problems encountered.

The participation to the meeting is mandatory for team members. If a member cannot attend, another one should report for him.

During a sprint, at the end of each day, every team member will update the sprint *burn-down chart*—a graph representing the amount of work left for the sprint (see **Error! No bookmark name given.**)—with the estimated *work left* on the tasks he worked on during the day. In this way it will be very simple to spot if the sprint is

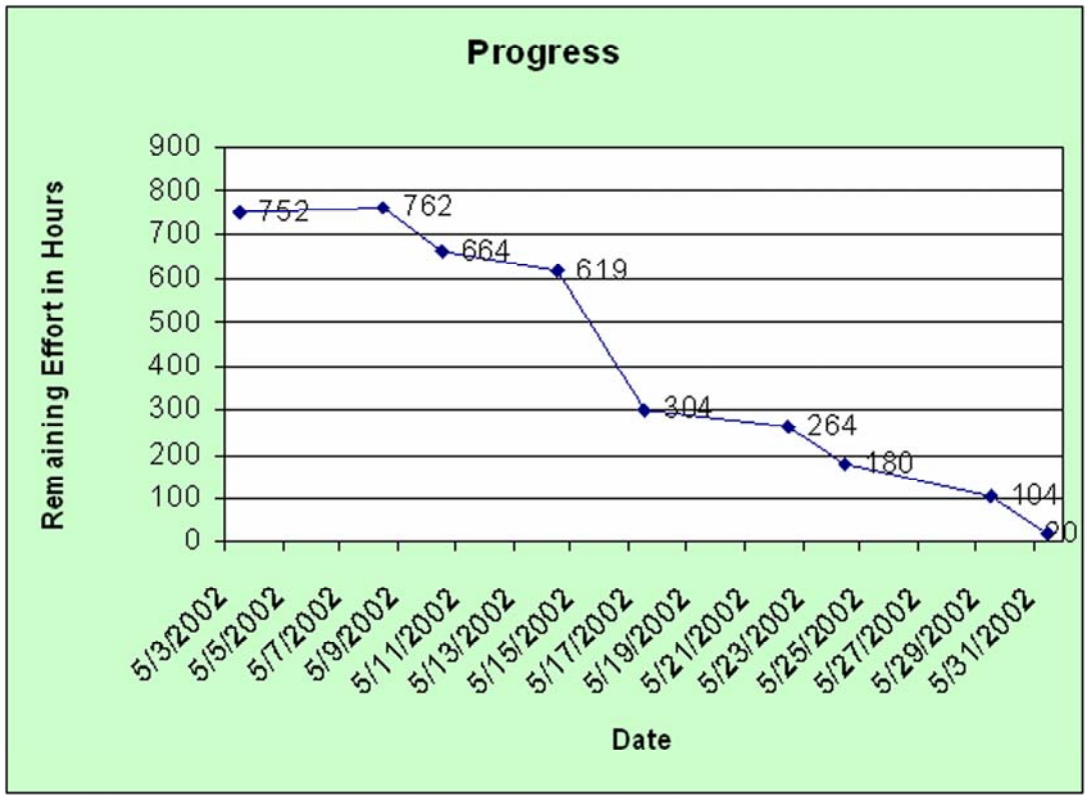

Figure 2: example of burn-down chart

going well, or there are delays requiring corrective action.

At the end of the sprint there is a *sprint review meeting* in which the team demonstrates to the product owner what it has achieved during the last iteration. After that, the product owner decides if the sprint goal has been met. This meeting is time-boxed to four hours.

Following the sprint review meeting there is a *sprint retrospective meeting*—time-boxed to three hours—in which the team and the scrum master talk about what went well during the last sprint and what can be improved in the next one. The product owner does not attend this meeting.

Finally, if something goes badly wrong during a sprint or the spring goal is not worthwhile anymore, then the scrum master or the product owner can call for an *abnormal termination of sprint*, in other words the Sprint is aborted, and action is taken to fix the problem and organize a new sprint with, possibly, a new backlog, and a new goal.

Implementing Scrum: Getting Started

Getting started, once there is a project, a product owner, a scrum master, and a team—from the technical point of view—is quite simple:

1. The product owner defines an initial a product backlog, a release plan, funding, etc.
2. The team along with the scrum master, and the product owner decide an iteration length (and stick to it)
3. Plan the first iteration: define the sprint backlog, the sprint goal, the tasks, and go

However, things don't always go smoothly, especially when there are people involved.

First of all, if the team is a newly formed one, there might be an initial period in which the team members will try to assert their power inside the team causing some problems of interaction between them until an equilibrium is found. This is a normal thing, and the product owner, and the scrum master have to resist the temptation—and refuse if asked to do so— of intervening to sort things out. The team has to find a proper balance without any external interference. In fact, any external help may prevent the team from self-organizing.

Another problem, very common in many organizations, is resistance to change. If you are trying to introduce Scrum in a organization that already uses another methodology—or no methodology at all—you are very likely to face the opposition of the ones that feel threatened by its introduction. In this case you may find useful to have a look at [8], in which the authors present a pattern language for introducing new ideas in an organization. I successfully used some of the techniques described in the book before knowing them as patterns [1].

In addition to the above problems, each role comes with its own challenges.

The scrum master has to resist the temptation to manage the team, even when, as occasionally happens, the team members ask for it—sometimes they request his intervention for issues that they should sort out among themselves.

The product owner, like the scrum master, has to resist the temptation to manage the team—which can organize itself differently from what he expects—and the temptation of adding more important work to the sprint backlog after a sprint is already started. If he tries to do so, the scrum master and the team should refuse. If the new features are so important that they must absolutely be there, the product owner can always call for an abnormal termination of sprint.

Another big challenge for a product owner is dealing with the prioritization of

backlog items. In fact, deciding what goes into a sprint and putting a priority on it, puts a big responsibility on him, who, if he is new to Scrum—or to iterative and incremental development—may find it very distressing. After all—this is the usual mindset—all the requirements have very high priority and all of them need to be satisfied in the final product. If this is the case, the team and the scrum master have to help him in defining proper priorities by asking questions, and, sometimes, making suggestions, so that he can clear up his mind and understand what the really important requirements to satisfy first are. This can be a very though job, but after a few sprints, if the team is able to deliver value at the end of each of them, his comfort level will rise, and everything will become much easier. In [1], even if it is not specifically about Scrum, you can find how my team and I dealt with this kind of problem in a real project I was involved.

In general, for someone in a managing position—project manager, technical leader, etc.—one of the biggest hurdles to overcome in implementing Scrum can be the perceived lack of control he may have on the project. For many people delegating authority, and trusting others is a real challenge due to the fear of loosing visibility, control, and, sometimes, personal power. Overcoming those fears is never simple, it usually requires hard work and time, and there are no sure recipes for success. The book by Mary Lynn Manns and Linda Rising can give you some good ideas about what—and how—to try [8].

As far as the team is concerned, Scrum is not for everybody: some people just don't like to be stripped of their title to become just team-members; the high level of interaction required may make the most introverted developers feel uncomfortable; and some developers are quite territorial with their own code, and don't like the idea of somebody else modifying or even reading it. These problems can be solved in several ways. What I usually do, is to try to convince the sceptics by understanding the issues they see with the methodology, showing them how these can be solved, and showing also what there is in Scrum that may be *personally* useful to them [8]—for programmers learning new technologies and techniques, usually, works a treat. It may happen that, despite your efforts, some people cannot simply be convinced even to try, in which case it would be better to exclude them from the team—so far I've never experienced this kind of situation.

Finally, some other problems can be caused by mistakes made due to lack of experience in introducing a new methodology in an organization. A couple of the most dangerous ones I've seen in practice are:

- Mandating the methodology from above
- Focusing too much on the process and not enough on the product

They are discussed in [3]. Here is a short summary of why they can be problematic.

The first one may happen if the project manager decides that he likes Scrum, and imposes it on the developers (and sometimes, on the product owner as well). The problem with this approach is that imposition very rarely works with programmers. So trying to impose them a new methodology can actually have an effect opposite to that which was intended.

The second one is typical of a team using a specific process for the first time. To a certain extent it is normal: after all, when you are learning something new you need to focus on it to see if what you are doing is right or wrong. However, when the team starts to spend more time on the process than on the product, it could be a sign of

something else going wrong.

That said, working in a well organized scrum project can be a very rewarding experience for everybody involved: the product owner has a better product earlier; the developers have more satisfaction from their job as well as the possibility of learning new things thanks to the cross-functionality of the team, and the involvement in every aspect of the project; the project manager has more control and visibility of what goes on in the project.

Scrum and Extreme Programming

Why this paragraph? After all there are several agile methods; so why compare Scrum with Extreme Programming (XP) [4]? The reason is simple: XP is probably the most well known agile method, and the first thing many people want to know when presented with another one is how do they compare.

However, comparing the two is a bit like comparing apples with oranges. Even if they share the values of the Agile Manifesto, they have a different focus. Scrum focuses on the project management side, leaving all engineering practices—design, coding, configuration management, testing, etc.—to the self-organization of the team, while XP focuses more on the engineering practices and doesn't provide any specific tools for project management. In practice, they are often used in conjunction, and there are also some methodologies built by merging the two. Have a look at [10] for an example.

Conclusion

In this article I introduced the basics of Scrum, the benefits of its adoption, and some of the problems—along with possible solutions—that may arise during its implementation.

If you want to learn more, in the references section you can find plenty of resources you can check—many of them available for free on the Internet. A good starting point are the ScrumAlliance [11] and the AgileAlliance [13] web-sites in which you can find several articles available for free about Agile Development in general and Scrum in particular. The ScrumAlliance one contains also several links to free software tools which you may find helpful if you decide to use Scrum in your organization.

Other resources on the web include also two very interesting Yahoo groups open to everybody: the Scrum Development [9] one, and the Agile Project Management [12] one, which you can use to ask questions to several agile development experts, and participate to various—sometimes heated—discussions.

Before trying to use Scrum in your project, apart from using the free resources described above, I suggest reading—at least—the book by Alistair Cockburn about agile software development in general [7], and the two books about Scrum by Ken Schwaber [15], and [14].

If decide that you really like Scrum—and are willing to spend some money—you can become a certified scrum master, by taking a two days course. You can find more details about it on the ScrumAlliance web-site [11].

Finally, I really recommend attending to conferences like the London XPDay (http://www.xpday.org) in which you can meet the experts in person, learn new tricks of the trade, and have a really good time while doing it.

References

[1] Asproni, G., *An Experience Report on Implementing a Custom Agile Methodology on a C++/Python Project*, Overload 64, 2004, http://www.giovanniasproni.com/articles

[2] Asproni, G., *Motivation, Teamwork, and Agile Development* , Agile Times Vol. 4, 2004, http://www.giovanniasproni.com/articles

[3] Asproni, G., *How to Shoot Yourself in the foot. In an Agile Way* , Overload 71, 2006, http://www.giovanniasproni.com/articles

[4] Beck, K., Andres, C., *Extreme Programming Explained: Embrace Change, 2nd ed.*, Addison Wesley, 2004

[5] Beck, K., et al., *The Agile Manifesto*, http://www.agilemanifesto.org

[6] Boehm, B. W., *Software Engineering Economics*, Prentice Hall, 1981

[7] Cockburn, A., *Agile Software Development*, Addison Wesley, 2002

[8] Manns, M. L., Rising, L., *Fearless Change: patterns for introducing new ideas*, Addison Wesley, 2004

[9] N.A., *Scrum Development Group*, http://groups.yahoo.com/group/scrumdevelopment/

[10] N.A., *XP @ Scrum*, http://www.controlchaos.com/about/xp.php

[11] N.A., *ScrumAlliance*, http://www.scrumalliance.org/

[12] N.A., *Agile Project Management Group*, http://finance.groups.yahoo.com/group/agileprojectmanagement/

[13] N.A., *AgileAlliance*, http://www.agilealliance.org

[14] Schwaber, K., *Agile Project Management with Scrum*, Microsoft Press, 2004

[15] Schwaber, K., Beedle, M., *Agile Software Development with Scrum*, Prentice Hall, 2002

