SCRUM
An Agile Model for Software Project Management

Décio Ferreira[1], Felipe Costa[2], Filipe Alonso[3], Pedro Alves[4], and Tiago Nunes[4]

[1] ei03037@fe.up.pt
[2] ei03041@fe.up.pt
[3] ei03044@fe.up.pt
[4] ei03083@fe.up.pt
[5] ei03101@fe.up.pt

Abstract. Scrum is an agile model of software project management. It follows a team-based incremental development, allowing a good control of the process by having short iteration cycles. This document intends to explain the methodology, from its origins to its very implementation.

1 What is Scrum?

Before we answer this question, another subject should first be introduced: Agile Software Development.

This is known as a "non-traditional" way to develop software, in which (and according to the Manifesto for Agile Software Development) client satisfaction is the highest priority, and it tries to achieve it by having early and continuous production of potentially shippable software, maintaining a close contact with the costumer.

Ken Schwaber, co-founder of Scrum, involved in the formation of the Agile Alliance and curiously also involved in the definition of traditional models such as Waterfall, describes this methodology referring to a rather amusing situation:

> *When someone asked me what I do for a living, I said that I help people build software in 30 days. And the guy looked at me and said "So, I don't have to wait 180 days to get what I* **don't** *want?" Yes, that's right, we'll give you what you* **don't** *want in 30 days.*

Having its principles fundamentally based in good management practices, Scrum is seen as an extremely agile and flexible methodology. Its objective is to define an iterative and incremental approach to product development or work management. It produces a potentially shippable set of functionality at the end of each iteration (normally with duration of 30 days). Focusing on team work, it thrives in maximizing cooperation and communication, and allows everyone to do their best and feel good with what they do, which later reflects in an increase of productivity.

Scrum is applicable to both small and large-scale projects. Trying hard to remove impediments to the development process, its main purpose is to correctly address the changing environment, constantly adapting to the chaos of interests and needs.

Being a wrapper for existent engineering processes, Scrum neither requires nor supplies any technique or specific method to the software development phase. It merely establishes sets of management rules and practices that should be adopted to ensure the project's success.

2 How did the concept come to life?

The Scrum methodology, developed by Ken Schwaber and Jeff Sutherland and inspired by Hirotaka Takeuchi and Ikujiro Nonaka's original ideas on rapid and concurrent product development, was born from the necessity to find an approach to non-traditional software development, in which and just like in a Rugby game, the team works as a whole trying to achieve it's objectives (a scrum is a team pack in rugby, where everybody in the pack acts together to move the ball down the field).

Some important marks:

- 1993 - Ken Schwaber develops an iterative and incremental framework at ADM;
- 1993 - Scrum methodology implemented for the first time by a team geared by Jeff Sutherland at *Easel Corporation*;
- 1994 - Jeff Sutherland defined Scrum at Easel;
- 1994 - Ken Schwaber and Jeff Sutherland refine Scrum;
- 1996 - IDX uses Scrum in projects with approximately 600 people;
- 1996 - Scrum is presented at OOPSLA (international conference on Object Oriented Programming Systems, Languages and Applications);
- 2000 - "eXtremme Programming" practices used together with Scrum (XP @ Scrum);
- 2001 - Ken Schwaber and Mike Beedle publish the first book on Scrum;
- 2003 - Scrum certifications;

3 How is Scrum software developed?

3.1 Specific vocabulary

Backlog	List of all functionalities to be developed, well defined and detailed at the beginning of the project. This should be sorted by priority of execution.
Sprint	Period of no more than 30 days, in which the project (or some of its functionality) is developed.
Sprint Backlog	Work to be developed in a Sprint, resulting in a deliverable product. It should be developed incrementally, relatively to the previous Backlog (if it exists).
Scrum	Daily meeting where progress and impediments to progress are evaluated.
Scrum Meeting Rules	Protocol to be followed in a Scrum meeting.
Scrum Team	The team working on a Sprint Backlog.
Scrum Master	Team element responsible for managing the process and conducting the Scrum meetings, normally software or system engineers. Despite being a manager he does not have authority on the other elements of the team. Self-management is encouraged.

3.2 Scrum Rules

So as to correctly build software using the agile method Scrum, it is necessary to follow a few execution rules, especially in what concerns the Backlog, the Sprint and the Scrum meetings.

The team should debate all the points that should be placed in the product Backlog, whereas the responsibility of sorting the list by priority of execution is of the Scrum Master alone.

As for the Sprint, which shouldn't last any longer than 30 days, a work team of no more than 9 elements should have a well defined objective, based on the Backlog. The Backlog should not be changed during a Sprint, with the exception of new functionalities that according to the Scrum Master have critical influence in the flow of the project and can be completed within the current Sprint's period. If the Sprint is likely to fail, a new Sprint can take its place, based on a new Sprint Backlog.

The Scrum meetings play a very important role in the project development. It is in those meetings that the Scrum Master has to get updates on the project

and find any causes for lack of productivity, so that he can effectively remove such impediments.

Meetings during a Sprint must be daily, at the same time and place, and shouldn't take longer than 30 minutes.

The whole conversation is about elements answering the Scrum Masters questions, namely:

1. What have you developed since the last meeting?
2. What were the main problems you found in the process?
3. What do you plan to have ready for our next meeting?

All elements should answer these questions, and based on those answers the Scrum Master should take immediate action (if necessary) to free the workflow from any existent problems.

3.3 Scrum Process

To start the Scrum process, the first thing we need is to define the team consisting of people who are assigned work that will compose the Scrum team. This team shouldn't have more than 6 to 9 members. If there are more members than manageable, break into multiple Scrums and make each Scrum team focus on one, self-contained area of work, involving all staff working in this defined area.

About the materials you must supply for each team, make sure you always have available flipcharts, flipchart markers, sticky notes, product backlog printouts and daily standup cheat sheet cards for each participant, so that everyone has access to all information and the means to prepare and understand each part of the Scrum process at anytime of it.

The next thing to do is to appoint the Scrum Master, since this is the person who conducts the Scrum meetings, empirically measures progress, makes decisions, and gets impediments out of the way of slowing or stopping work. This is often the engineering or marketing manager for this product or system area. The Scrum Master is in charge of asking all team members the three previously mentioned questions.

He is the one who must be able to make immediate decisions and resolve work impediments as soon as possible, not to extend the discussion time much further. The Scrum Master is who identifies the initial backlog, which is all of the work that is outstanding for a product area, both immediate and well-defined, and long terms and visionary.

To identify the Backlog the first thing to do is to list all known work to be done and group it into increments that should take no more than 30 days. If there are areas where work is volatile or cannot be fully defined for up to 30 days, an increment for a known horizon must be established . After that, we need to list all outstanding work to be done and prioritize all collected items in the list. When finished, the backlog should be signed up for by team members and since that day, only this backlog is worked on during this Sprint for each area.

From this point on, to conduct work collaboratively and check-in on progress within the team, you only need to self-organize, begin the work assignments and conduct the Daily Scrum Meetings. It's vital for this process to work to complete work assignments rigorously based on Sprint Backlog work remaining.

That for, you must establish and conduct Daily Scrum Meeting where the team meets and updates each other about what's going on. It provides a daily focus on the work being done. First of all, make sure the meeting occur always at the same time and place, avoiding overhead of finding a place daily and also avoiding overhead of team trying to figure out where and when is today's meeting. Remember that each meeting must take no longer than 30 minutes. During this time the Scrum Master plays its role on asking the questions and making all necessary decisions. All discussion other than replies to the 3 questions defer to later meetings.

At the end of the sprint, there must be a Sprint Review and Demo meeting. To conduct a Sprint Review and Demo meeting elect a spokesperson to guide it and conduct the meeting in order to get some questions answered and register the group-wide Sprint retrospective:

1. What is the potentially shippable increment (Demo)?
2. What did we complete of our Sprint Backlog?
3. What is the feedback from our Product Owner?
4. What did you notice happening in your group in completing your Sprint?
5. How did that feel?
6. What does that tell you?
7. What could you apply to work better with next Scrum Sprint?

You might also consider some additional facilitator guidelines as providing a flipchart pad to each team to capture their sprint backlog, sprint review and demo meeting purpose and any other information that supports their demo. Capture all results digitally on camera to review and confront if needed, perfectly explain the rules for the process to run better such as that the team must work together, that everyone must have work in the Sprint and each team must demo something at the end of the Sprint. Don't allow PowerPoint slides since it's a waste of time making them and explaining them in short time meetings. Focus that the team must complete their Sprint Planning with a Sprint Backlog displayed on a flipchart and that there is no predefined roles on the team, since the goal is self-organization rules!

4 Practical Applications

Ken Schwaber tells us that some common answers to the question "Why have you decided to implement Scrum" are "we have nothing left to lose" or "it couldn't get any worse". As already seen, the traditional methods do not deal very well with the associated risk in developing a project, which makes it common to have organizations looking for a "salvation" when they realize they are on their way

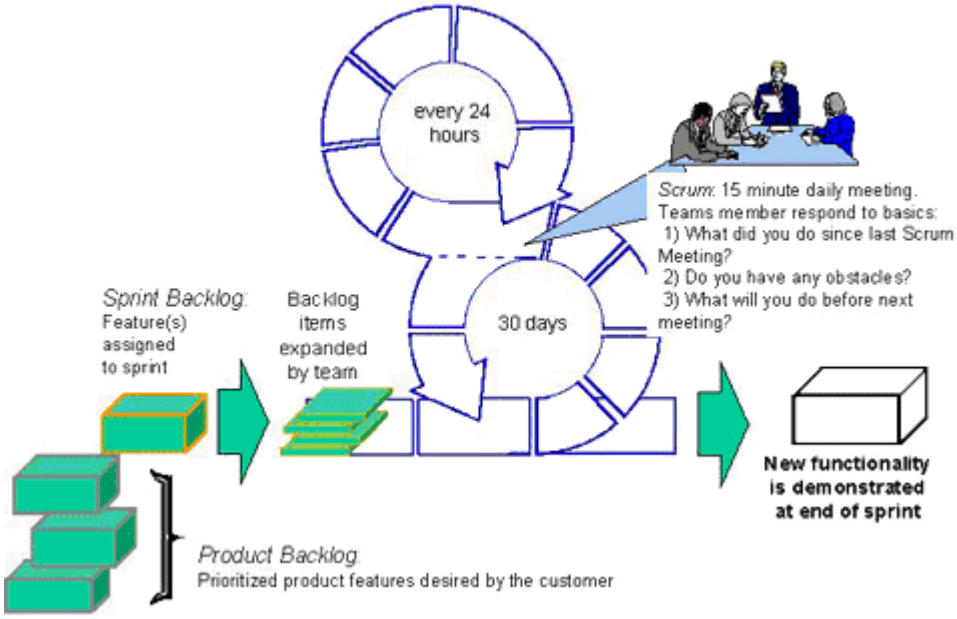

Fig. 1. Description of the Scrum process

to failure. Scrum manages to regularly evaluate the environment and adapt to it, minimizing risks and producing the functionalities it can at the moment.

However, only one third of the organizations that try to implement Scrum as a development model manages to succeed. As a matter of fact, the traditional development structures somehow hinder an easy transition for agile methods, due to the habits they instil. For example, while traditionally a project manager commands and controls development teams, Scrum is based on the capability of a team to self-manage. Actually, the project manager is given another designation (Scrum Master) to somehow emphasize the fact that he has no authority on teams. Another example is the fact that intervening elements in development have, in Scrum, varied functions (to maximize the teams productivity), instead of having defined and limited functions. It takes a great deal of effort to let go of these habits and achieve a rather rough twist in terms of methodologies. As Schwaber says, Scrum is not a "silver bullet"; it is indeed a lot of hard work.

After overcoming these difficulties (which is absolutely possible, companies like Philips, Cisco Systems, Siemens, Nokia, Microsoft and Sun Microsystems already have experience in using this method), perseverance is important. Since each Sprint aims at implementing a certain set of requirements, what usually happens to an organization that starts using Scrum is to fail many of the requirements in the first iteration. Several reasons are behind this: lack of confidence, team members do not know each other very well, technological difficulties... Af-

ter the second iteration, it is already normal to observe an approximation to the predefined goals and eventually, when already proficient in the process, the team achieves notable accuracy.

According to some studies, during the first months implementing Scrum, organizations manage to fulfil about 5 requirements at a cost of 70,000 Euros (50,000 British Pounds), with an average of 100 defects in production. One year later, they usually double the completed requirements for the same cost and number of defects. In the second year, they fulfil about 14 requirements for the same price, but now with an average of 5 defects in production. These numbers show the evolution of the new methodology's implementation process. The initial concern is focused on obtaining the desired relationship between teams and customer and getting the team to interact as needed. Only after mastering this do we try to improve the inherent engineering aspects of development, reducing errors in production.

It is known that agile processes greatly increase productivity. In terms of Function Points (used to "measure" an application in terms of its functionalities) and while the industry standard is of 2 Function Points per person per month, an (optimized) agile process reaches the 77 Function Points per person per month. Despite this impressive inflation, workers' quality of life is not at all negatively affected. As a matter of fact, the collaborative character of Scrum's philosophy, where cubicles and "orders from above" are totally abolished and communication is encouraged as the means to problem solving, greatly increases team morale. People go to work with high expectations for the day and create a vibrant environment. It has been said that it is possible to identify the working methods of an organization by simply evaluating the environment inside: if the noise level is low, it is certainly not using an agile method.

It is common to implement other methodologies in parallel with Scrum. For example, eXtreme Programming is frequently used to fill in possible gaps. This usually concerns quality, because of the speed at which software is developed (XP allows more control since a functionality is only considered ready when approved in the unit and acceptance tests). This way it is possible to further improve the development process in an organization.

5 Advantages versus Disadvantages

Scrum has its own advantages and disadvantages, like any other software production methodology or process. It can co-exist with other methods in order to fill in existing gaps, but obviously never reaching total perfection.

Software production is not always based on perfection. Customer satisfaction, by both rapidly delivering and fulfilling deadlines and accompanying him throughout the project's evolution (which implies that he sudden and unexpectedly changes his mind), is generally more important. Also, we cannot ignore that the customer is (normally) only a user and merely wants the software to meet his needs, with no interest in how it makes it happen.

As previously stated, the agile nature of Scrum will make it improve productivity. This does depend, though, on the Scrum Masters' motivation and strictness. They must therefore have a deep understanding of the process and implement it as rigorously as possible, so as to get the highest possible productivity.

In terms of innovation, Scrum supplies a bottom-up "continuous, fast reengineering", allowing massive sudden changes to the project without great damage in terms of time and costs. It makes this possible by turning the project into a set of small problems, "pieces" of a bigger problem, much easier to manage than if we tried to approach it in its integral form.

Normally, in complex projects, the interface between the diverse modules is too complex due to the high degree of isolation of the involved members. In Scrum, getting teams to work together and communicate makes work integration much easier, since everyone knows what everyone is doing.

Another advantage is the short intervals of time between project presentations to the customer. This enables customer to have a closer look at the project's evolution and get frequent feedback concerning the product's functionalities. Soon, a relationship with the customer is developed, confidence is built and knowledge grows. Obviously, the communication between team elements improves substantially and all the successes throughout the process are shared, which contributes to creating a culture where the people involved really hope for the project to succeed, highly motivating the team. This kind of trait is very important in present organizations. This method can be applied to any another process, however, since it is merely a set of values, practices and principles.

Unfortunately, as with any other method, Scrum does present some problems that are difficult and perhaps even impossible to surpass, due to its very nature.

Scrum is a method that demands an "on-hand" management. This implies that the manager has to constantly be willing to make modifications, in order to promote assistance and to help teams have success by removing impediments, which requires constant monitoring.

Development teams still have to operate adaptively in a complex environment, using inexact processes. This method requires that the management give authority of decision to Scrum teams. The managers must let teams take their own decisions, allowing these to fail when doing so, if necessary. All this, although allowing a bigger evolution and nimbleness in future projects (if they actually learned something with their mistakes), does imply costs, both in time and money. However, allowing short-term damage might reflect in benefits in the long run.

Beyond all these factors, it still has to be considered that scrum is a new and different method and that people, in a general way, are resistant to changes. This can generate an initial discomfort with the method, require an adaptation phase and, in the worst case, some people might not at all adapt to the rhythm or methodology of Scrum. Since it usually implies a big change in the organizational culture, its implementation must be carefully analyzed. After the adaptation process, another frequent problem is the possibility of some members or even sub-

managers not to feel comfortable with the responsibilities that Scrum demands, or management feeling uncomfortable with the lack of visibility they have.

5.1 Comparison with other models

	Waterfall	Spiral	Iterative	SCRUM
Defined processes	Required	Required	Required	Planning & Closure only
Final product	Determined during planning	Determined during planning	Set during project	Set during project
Project cost	Determined during planning	Partially variable	Set during project	Set during project
Completion date	Determined during planning	Partially variable	Set during project	Set during project
Responsiveness to environment	Planning only	Planning primarily	At end of each iteration	**Throughout**
Team flexibility, creativity	Limited, cookbook approach	Limited, cookbook approach	Limited, cookbook approach	**Unlimited during iterations**
Knowledge transfer	Training prior to project	Training prior to project	Training prior to project	**Teamwork during project**
Probability of success	Low	Medium low	Medium	**High**

Fig. 2. Comparison table with other methodologies of the sector

In the table we can see that relatively to other methodologies of the sector, Scrum is the only one that allows limitless flexibility and creativity of the teams through iterations (meetings and successive innovations in the project everyday), which does characterize this method. Knowledge transference, like already said, is made throughout the project in joint team work and sharing of reached goals. Achieving an efficient risk assessment, Scrum enables a high probability of success.

It is possible to make an analogy of the future of agile methodologies with the object-oriented methodologies for software development. In the same way that some object-oriented methodologies appeared and competed between themselves, several agile methodologies also appeared. The need for standardization arose, resulting in UML, currently a standard in the software industry. In the same way, in the future, we might see agile methodologies fuse, especially around XP, which is more widely accepted. There is, for example, a movement for the joint use of XP and Scrum (XP@Scrum). XP would be used in the development phase and Scrum for project planning and management. The integration of the two methodologies would be relatively simple, since they share some characteristics (like the need for customer presence, small releases and encouraging changes, necessary to address stakeholders' actual requirements).

Summing up, Scrum approaches software development in an empirical way, much like a flock of migrating birds: they evaluate the weather and decide it

is too cold to stay where they are; deciding to leave, they analyze a series of conditions to define the direction they will follow; once they reach a place that is good enough, they will stop there, even if it is half the way they initially thought it would be.

If the costumer lives for more than a month, his mind is going to change. If the project lasts for more than a month, requirements will change and the development will have to adapt to those changes if it wants to embrace them. It is of absolutely no use to spend time, effort and money exhaustively planning a project in its beginning if the world is not supposed to freeze while we work on it. Neither is it useful to place people, who are supposedly part of a team, working in cubicles and reporting to each other whenever they're asked to (which could be never). Software development is nowhere similar to car manufacturing. Software is never thoroughly defined in the beginning of the process, and there is no production line where people further down that line don't need to worry about what others did before them. Why treat software development as a defined process, then? Scrum is therefore summed up to two words: common sense.

References

[November 2005] http://www.controlchaos.com
[November 2005] http://www.jeffsutherland.com
[November 2005] http://www.presidentekennedy.br
[November 2005] http://www.scrumalliance.org
[November 2005] http://www.scrum-master.com
[November 2005] http://hubert.blogs.com/dahdidahdi_dadadihda/