Moving Back to Scrum and Scaling to Scrum of Scrums in Less Than One Year

Rafael P. Maranzato
Universo Online S.A.
Dept. of R&D
Sao Paulo, SP, Brazil
rmaranzato@uolinc.com

Marden Neubert
Universo Online S.A.
Dept. of R&D
Sao Paulo, SP, Brazil
mneubert@uolinc.com

Paula Herculano
Universo Online S.A.
Dept. of R&D
Sao Paulo, SP, Brazil
pherculano@uolinc.com

Abstract

We report on the experience of re-introducing Scrum in a project team that had previously failed to adopt that agile method. We explore the reasons we believe that caused the failure and explain how we approached the team to uncover them. Then, we describe our strategy to avoid incurring in those problems again and to take the team to a higher level of productivity, quality and personal satisfaction. We also present the motivation and the actions taken to go further and scale this scenario to multiple feature-oriented teams using Scrum of Scrums. All these changes occurred in less than one year.

Categories and Subject Descriptors K.6.1 [*Project and People Management*]: Management techniques

General Terms Experimentation, Management, Human Factors

Keywords Scrum, agile, scaling Scrum, Scrum of Scrums, experience, cultural change

1. Introduction

This paper is based on our experience in the Research and Development (R&D) department of *Universo Online* (UOL), the largest Internet portal in Brazil. UOL was launched in 1996 as an Internet Service Provider focusing on content providing and basic Internet services, such as chat and e-mail. Most of the projects were small and short lived; services were launched and few of them evolved significantly after that. Around 2006, following the company's IPO in the Brazilian stock market, it expanded its portfolio to offer new services that would be managed by independent business units. Most of those services were meant to compete with incumbent Internet companies in Brazil and included an online marketplace, a price comparison shopping, a sponsored links platform, among many others.

In the Research and Development (R&D) department, significant changes also happened during this period. In the first few years there was no formal process and each team decided how to develop and deliver software to the company. In 2000 we started using the Rational Unified Process (RUP) [3]. At first, we used most of the documents and templates provided by RUP, but along the years we adapted our process and focused only in core artifacts such as Vision and Use Cases [2]. Although we felt that this process was inefficient, until 2006 it was compatible with the type of projects that we were assigned—simple and fixed scope, relatively short duration (at most three months) and no fixed team.

Around 2006 we began to realize that this process was becoming a burden given the new business needs of the company. The newly constituted business units demanded continuous effort in the development and evolution of their software products. Soon the company decided that these products should have fixed teams, but at first, these teams were composed only of Java developers. Other skills necessary to build the software, such as testers, webmasters, data administrators and system administrators, were still allocated by demand from the functional teams they belonged. This situation exacerbated the inefficiency of our process and we began to actively discuss how we should change the way developed software.

In the beginning of 2008, we had come to the conclusion that Scrum [5] was the best choice for our scenario and we started its implementation in three pilot projects. One of the authors had the opportunity to be the first Scrum Master of the company and was involved in two of the pilot projects. One of them was the evolution of a very complex software, an online marketplace for buying and selling products. The other project was creating a new tool for managing complex sets of data and metadata, targeted to internal users. Although these projects were very different in their scope, stage of development and relevance to the company, Scrum proved to be an excellent fit for both.

The third pilot Scrum project was also very ambitious and still different in purpose from the other two: its goal was to rewrite a very large system, replacing an old and limited platform with a new and efficient one, while maintaining all functionalities. However, Scrum was not successfully implemented in this project. Viewed from the outside, the team seemed just not able to make progress. As we later learned, they were lost in pointless technical analyses, endless discussions and personal conflicts. Initially expected to be completed in nine months, the project was already running for almost one and a half years when management concluded that Scrum might be causing the delays. In order to meet a new deadline, they returned to a more traditional project management approach, abandoning Scrum. The project was actually delivered a few months after the new deadline, but not without much distress, extra hours and removed functionalities. After the release one, a group of managers with more experience with Scrum, including two of the authors, was involved in the project and to put the team back on the tracks with Scrum. And in less than one year, we have scaled Scrum in four feature teams. This is the experience that we will describe in this paper.

SPLASH'11 Companion, October 22–27, 2011, Portland, Oregon, USA.

This paper is organized as follows. Section 2 presents the team in which we based our study and its first experience with Scrum. Section 3 describes how Scrum was reintroduced to this team and the new approaches that we took. Section 4 draws conclusions on our report. Some Scrum terminology can be found in Appendix 5.

2. First Agile Experience

As soon as the company decided to adopt Scrum in early 2008, the next step was to provide training in agile fundamentals for all professionals involved in product development, from software engineers to business specialists. Beside the pilot product teams, we created interest groups gathering Scrum Masters, Product Owners and other technical skills. The new product teams were at the spotlight of the company and Scrum was the most spoken word in the environment. Everyone wanted to learn or teach something. The groups were infected by agile feelings. People did not need to use boring processes and corporate tools anymore. They were allowed to choose the tools they considered appropriate because, after all, the Agile Manifesto said "Individuals and interactions over processes and tools [1].

As we mentioned before, one of the pilot product teams had to rewrite a system built in an old platform. The requirements were well known and there were not many business risks involved, but there were significant technology risks. The team was formed by professionals that had never been in a product team before. Instead, they came from different departments, organized according to their technical skills. This should not be a problem if the involved departments had common goals, but that was not the case. The department managers gave directions to their professionals according to their old way of working, without considering that things had changed. For example, in the old methodology there was a person responsible for testing. With Scrum, there is a testing role inside the team, which could be performed by a test engineer or a software engineer. But this did not sound correct to the functional departments, and they wanted to create rules for their members. For instance, each funcional manager involved wanted to define what the concept of *Sprint Done* meant, according to their own criteria.

As we should expect, team members coming from different functional departments did not agree with each other. Each was following a different agenda, according to the objectives of their departments. This led to very long discussions during the sprints, mostly during the *Daily Scrums*, which almost always exceeded the recommended 15 minutes. There were also lots of blaming and unfruitful discussions in Retrospectives. Since people were not aiming at the same target, which should be delivering business value in the form of working software, they could not agree on many topics. One example of disagreement was related to the use of a new technology in the project. One of the functional departments decided that the project was a good opportunity to validate a new technology they wanted to employ in their own projects. Since most of the developers of the team reported to that department, they were instructed to study and use this technology. This decision slowed down the project for about three months. When the Product Owner noticed this and questioned the head of that department, he answered that, according to Scrum, this technology decision was a prerrogative of the team, so there was nothing wrong with their choice. After a few months debate, the issue scaled up to the head of R&D and the team was instructed to abort the use of the new technology.

Sprint after sprint, most of the user stories were considered *Not Done*, due to the conflicting definitions that each department had on the concept of *Done*. This would accumulate more work to each subsequent sprint, generally to make adjustments that would not generate value to the end product. As a consequence of all these unsuccessful sprints, the conflicts inside the team were becoming more frequent. Discussions on sprint velocity and the format of meetings were also among the top discussions and disagreements. Surrounding all conflicts, or adding one more, there was the estimated time to accomplish the project, which was uncertain yet. And all of these problems became due to Scrum, so they were considered all *Scrum's fault*. By that time the mindset was: "If we want to deliver this project we should come back to the traditional model because we know we have problems but we deliver projects. And then the decision was made: the team was not required to use Scrum anymore and then could be free again. The release one was delivered but there was a directive to use Scrum again, since it was being successfully used in other teams in the company.

3. Moving Back to Scrum

In the beginning of 2010, before returning to Scrum, we talked a lot with the most experienced people in the team to discover what they were expecting from that return and what important lessons could be learned from the past. After some talks we could see they were very focused on some rules, such as the definition of done, the opposite of Scrum and agile values. At this time, over the regular evolution of the system, there was a business need to integrate that system with other services of the company. We could see that it was a different backlog, and a good approach would be to split the team as we split the product backlog.

To start the process, we reintroduced Scrum to the team focusing on the values, like commitment, transparency, teamwork and so on. and we had to change one important concept of the majority of the team: that it was possible to have more than one team working in the same software components. So, we broke the team and the backlog by feature, creating feature teams [4]. Another important change that we made was to add three new members to the team that had experience in working with Scrum. Besides their technical skills, these three professionals were easygoing people who could aggregate positive attitudes during the sprints.

One important concept that we have empathized and that continues nowadays is that we could break the backlog into two or more feature teams but everybody were part of one team. No matter what happens, we were one team. It was very important because it is not possible to have team competition, velocity comparison, owner of software components, bugs per team and things that could be harmful to teamwork.

With this preparation, we started these two sprint plannings on the same day and we invited people from the other team to be at the sprint planning, to see what features would be created. We also invited them to the sprint review to share what each team had produced.

We also asked team members to help and participate of some decisions, like suggestion to the backlog (including features to the product and technical improvements), test infrastructure and other question that in the past were concentrated at the management level or with one or two team members - the others were just informed of those decisions, and we think that approach is opposed to agile values, since we need people commitment.

After six months of these two teams running sprints quite well, we could see the good results of creating features teams. At the same time, the results of the business and the increase of revenues and users in the product, made the business owners and the IT management consider creating more teams to focus on specific backlogs. It was good because we could see that our approach was bringing results, but, of course, we also faced some problems.

At this moment, one of the problems was the need of hiring more team members and how to put them in a system that has a lot of business rules and deals with money. We agreed that it would be better to hire good people and wait to start new feature teams instead of just increasing the team members and starting new

teams. We also have proposed to transfer people from other teams inside the company because they were used to developing with the company procedures and rules.

As we were hiring, we increased the existent teams before creating the third and the forth one that we have identified as priority. In parallel, we created the backlog for the new feature teams. Another decision that we made was not to start the forth team at the same time as the third one: we knew that we would learn from that division and face some impediments. When we had people and the backlog ready to start, we initiated the third team and we found some issues. The most relevant were:

- Teams per release: we observed that having more than 3 teams per release would be hard to manage considering the number of functionalities, bug fixes and database modifications.
- Merge instead of one build: considering the previous issue, we prefer to maintain two branches of code. One to the teams A & B and other to teams C & D, for example. As A & B deliver, teams C & D merge the code and A & B start a new branch. The opposite occurs when C & D deliver. This example is in Figure 1.
- Sprint length: before starting the third team, we spend 3 weeks in the sprint. We agreed that to add one more week would be easy to fit the sprint length during the days of the week.
- Develop environment: each team needs a proper environment without sharing with another team. The Quality Assurance (QA) environment is unique for all the teams because it reflects the production environment and when features from different teams run in the same build.

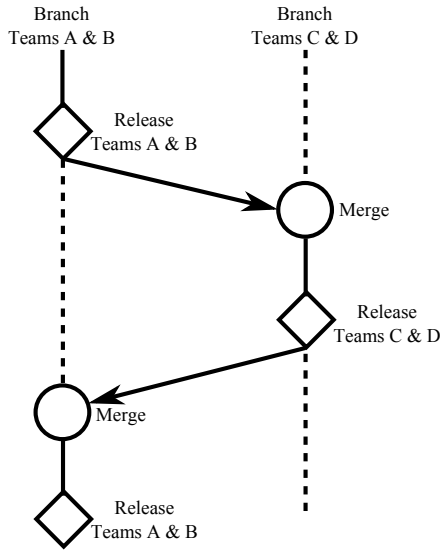

Figure 1. Branches between teams

These impediments and others helped us to understand how to implement a good Scrum of Scrums. We could see it would be very important to improve the communication among ScrumMasters, Product Owners and all the teams. So, as we had some rituals in the Scrum, we added some meetings to the team agenda:

- All Product Owners and ScrumMasters: there is a meeting every week with all Product Owners and ScrumMasters to talk about features that have impact on the other teams or about subjects that are relevant for everyone. We spend around 90 minutes on this meeting.
- Mega Planning: just after we finish the sprint planning of the teams, that is usually every other Monday, we talk to all the team members about the stories that each team has chosen. These meetings are very important because a member of one team can advise or warn the others and people can see the impact of these stories among the teams. It is important to emphasize that we start this meeting talking about the schedule ahead for everyone to know the deadlines and the relevant dates during the sprint. We also address generic topics that are important to most of the team and team members. We spend less than one hour with this meeting.
- Mega Daily: in the weeks when we do not have sprint plannings, there is a meeting that is similar to the Mega Planning. The main difference is that we focus on the status of development of new features and it lasts around 30 minutes.
- Knowledge Sharing: once or twice a month, usually on Fridays, we have some presentations about training or conferences people attend, relevant features or best practices that we consider important for everyone. We also share experiences from some teams to the others.
- Members of new teams: we do not create new teams with new people in the organization. Basically we add new employees into the existent teams and we create new teams with the old ones. It motivates everybody because people know that there will be opportunities of creating and learning new things. It is also important to knowledge sharing.

One example of this meeting during the weekdays is the listed schedule in Table 1. As we can see, we add more points of communication to our Scrum framework. We believe that improving communication and creating these rituals we can synchronize the development of the features in each team with the whole system. We think we can add two more teams to that agenda, but more than that would probably be hard to administrate.

Another issue that we think we could have is that although our mega meetings are productive, it is hard to find rooms that are big enough for everyone. When this happens, we will have to try other approaches like virtual meetings or we may choose teams' representatives to attend these meetings and pass on all the information after the daily Scrum. But we are not comfortable with this situation because we think this is an important channel of communication that we have nowadays, especially with mega plannings and mega dailies.

If the team, in the beginning of this process, was very distrustful about splitting the job into feature teams, nowadays they have learned how to work in this way and are contributing to scale our development process. One example is related to improvements in our development environment and refactorings that are necessary to facilitate working in parallel. It is important to remember that the system was created with a vision of one or two teams working in parallel, but the business is growing fast and we cannot stop the market – we need to keep up with this growing. So, people started to suggest refactoring in components, builds and other things because they are convinced that it is the responsibility of all the feature teams - and they are happy with the results of their work.

After 10 months of returning to Scrum, we checked with our business clients how they evaluated these changes. We did had a meeting - similar with to the Scrum retrospectives - and the balance was very positive: they were satisfied with the feature teams and they asked us how to create more teams, because the backlog is enormous and it is impossible for only one or two teams only to develop it. Another important improvement that we had was in the relationship between the R&D and business people. Before Scrum there was a lot of misunderstands and competition between

Monday	Tuesday	Wednesday	Thursday	Friday
Planning A & B	Mega Planning			Knowledge Sharing
Mega Daily		Review C & D	Release C & D	Retrospective C & D
Planning C & D	Mega Planning			Knowledge Sharing
Mega Daily		Review A & B	Release A & B	Retrospective A & B
Planning A & B	Mega Planning			Knowledge Sharing

Table 1. Basic Schedule

areas. Nowadays this relationship is much more trustful and we are continuing trying to improve it.

As explained before, the relationship between the teams and the environment had an improvement if you compare it to the times without Scrum. We can also observe that the business area is happy with the new methodology. Besides that, we have had an increment in the team velocity.

When we started the second team and split the backlog, we told the teams to create its own velocity and we did the same to the third and fourth ones. After that, we conducted an experiment to compare the story's sizes of teams and we observed that the estimates were very similar among these teams. It can be a coincidence but we believe that is explained because some members worked in two or more teams and probably influenced this fact.

Considering that the estimates are similar, we could see that we have had an improvement in team velocity meanwhile we were adding more members and creating new teams. In Figure 2, we can see that our velocity was growing month by month (line points) and our productivity too (line points per person). There is a peak in October but we consider that an abnormal event because it is out of the trend.

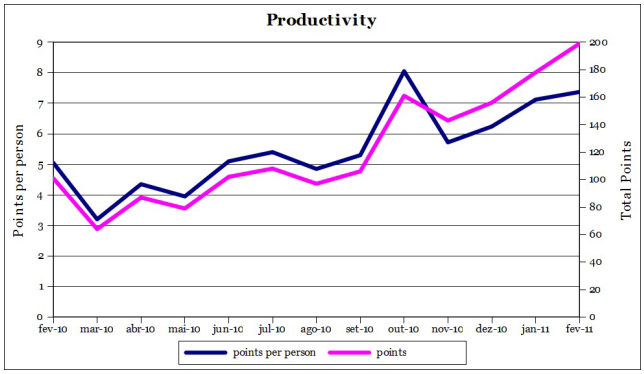

Figure 2. Productivity

As you can see, we have made a lot of modifications with this team in less than one year. Firstly, we had a cultural change while the management and the methodology were modified. Secondly, we changed their vision of the product so that we could have a lot of feature teams but we are just one big team. In addition, we changed the methodology in the product development using Scrum, focusing on values like commitment and transparency. And finally, we scaled Scrum creating more feature teams, using Scrum of Scrums.

Nowadays, our main challenge is to maintain our vision and values meanwhile we continue creating more feature teams. We believe that we will face some problems and issues but we also believe that by using an approach similar to the one that we have used until now we will be able to do that.

4. Conclusion

This paper described how we managed to implement Scrum in a team that had previously failed to adopt the agile culture. We also showed how we were able to scale this team to multiple feature-oriented teams, using Scrum of Scrums. All this transformation was done in less than one year. We found that some key factors for our success were the early identification of the causes of the initial failure of Scrum adoption, the allocation of professionals to Scrum roles according to their abilities and the addition of new professionals with good team skills and experience with Scrum.

Regarding the need for scaling the team, we demonstrated that separating teams by feature was the best decision in order to focus on the main priorities for the evolution of the product and to avoid conflicts in backlog management. We concluded that it is very important to create each new team with at least one professional with good technical skills and knowledge of Scrum, backed-up by an experienced ScrumMaster and a Product Owner. The practices we employed to deal with multiple teams – such as mega plannings and mega dailies – were crucial to keep the information flowing through all teams and maintain the concept that everybody belongs to one big team. Moreover, techniques such as parallel version control, continuous integration and carefully planned deploys were important to keep the quality of the work even with many new members being added in a short period of time.

5. Appendix: Scrum Terminology

This section is based on [6].

- Daily Scrum Meeting: A stand up fifteen-minute daily meeting for each team member to answer three questions:
  1. "What have I done since the last Daily Scrum meeting? (i.e. yesterday)"
  2. "What will I do before the next Daily Scrum meeting? (i.e. today)"
  3. "What prevents me from performing my work as efficiently as possible?"

  The third question refers to the Impediments and it is assigned to the ScrumMaster.
- Impediments: Anything that prevents a team member from performing work as efficiently as possible is an impediment. The ScrumMaster is charged with ensuring impediments get resolved.
- Product Backlog: The product backlog (or "backlog") is the requirements for a system, expressed as a prioritized list of product backlog Items. These included both functional and non-functional customer requirements, as well as technical team-generated requirements. While there are multiple inputs to the product backlog, it is the sole responsibility of the product owner to prioritize the product backlog.
- Product Backlog Item: In Scrum, a product backlog item ("PBI", "backlog item", or "item") is a unit of work small enough to be completed by a team in one Sprint iteration.

Backlog items are decomposed into one or more tasks. Some practitioners represent backlog items into User Stories.

- Product Backlog Item Effort: Some Scrum practitioners estimate the effort of product backlog items in ideal engineering days, but many people prefer less concrete-sounding backlog effort estimation units. Alternative units might include story points, function points, or "t-shirt sizes" (1 for small, 2 for medium, etc.).
- Scrum Roles: There are three essential roles in any Scrum project:
  - Product Owner: A person that represents the customer's interest in backlog prioritization and requirements questions.
  - ScrumMaster: A facilitator for the team and product owner
  - Team: A group of people responsible for constructing and delivering the product. The team members are usually a mixture of software engineers, architects, programmers, analysts, QA experts, testers, UI designers, etc. This is often called "cross-functional project teams".
- Sprint: An iteration of work during which an increment of product functionality is implemented.
- Sprint Backlog: Defines the work for a sprint, represented by the set of tasks that must be completed to achieve the sprint's goals, and selected set of product backlog items.
- Sprint Planning Meeting: The Sprint planning meeting is when team and the Product Owner negociate and agree on what will be delivered in the end of the Sprint.
- Sprint Review Meeting: A Sprint Review is a meeting where the team demonstrates working software corresponding to project backlog items they have completed in a given Sprint.
- Sprint Retrospective Meeting: The Sprint retrospective meeting is when the team discusses what went well and they should keep doing and what to improve in the next Sprint.
- Velocity: The velocity is calculated based on how much product backlog effort a team can handle in one Sprint. This is based on the average of previous Sprints and it should assume that the team composition and the Sprint lenght are kept constant.

Acknowledgments

First of all, we thank the *Phoenix* team, which we analyzed in this report. All team members were completely available to our interviews and provided very sincere and insighful opinions on the project. We also thank Marcio Drumond, head of the Department of R&D, for supporting this research and revising our drafts.

References

[1] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. Manifesto for agile software development, 2001. URL `http://www.agilemanifesto.org/`.

[2] M. Fowler and K. Scott. *UML distilled - a brief guide to the Standard Object Modeling Language (2. ed.)*. notThenot Addison-Wesley object technology series. Addison-Wesley-Longman, 2000. ISBN 978-0-201-65783-8.

[3] P. Kruchten. *The Rational Unified Process: An Introduction*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3 edition, 2003. ISBN 0321197704.

[4] C. Larman and B. Vodde. *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. Addison-Wesley Professional, 1 edition, 2008. ISBN 0321480961, 9780321480965.

[5] K. Schwaber and M. Beedle. *Agile Software Development with Scrum*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001. ISBN 0130676349.

[6] V. Szalvay. Glossary of scrum terms @online, mar 2007. URL `http://www.scrumalliance.org/articles/39-glossary-of-scrum-terms`.