Aalto University
School of Science
and Technology

CSE-C2610
Software Project

Learning Diary

Marisa

Samuel Marisa 79770K
Neronet Team 11

March 5, 2016

Contents

1 Scrum game

This entry was written on October 8, 2015 a week after my participation in the scrum game with my team. Here I'll answer the questions put forward by the organizers:

What were the most important things that you learned? Discuss critically what you learned during the game (and mention if you felt you did not learn something)

- I learned through practice that perhaps the most important function of sprint planning for the development team is to discuss and unify understanding regarding the work items that are identified in the sprint backlog and prioritized by the PO.
- Straightforward, effective communication is key to success in almost every part of Scrum.
- Preparing to work without permission to discuss emphasized the importance of planning and careful allocation of responsibilities. This is relatively easy in projects where the future work content is easy to understand and thus plan such as in the case of building something everybody has a pretty decent understanding of with Legos. Repeated sharing of understanding and redirection of plans at short intervals is likely much more important when the work is less understood.
- I got to experience how the scrum master can behave in practice and how the interaction between him/her, the PO and the rest of the team can work in a pretty generic setting.
- I relearned how easy it is to misunderstand requirements and thus waste time and how tricky asking good questions about requirements can be. Prototyping is a very effective way to alleviate this problem as it often works as a great tool for communicating requirements, priorities and ideas for potential solutions.

Do you think that the game will help your project succeed? How?

I am certain that the game experience will help our project succeed because:

- Of the important learning I already explained above.
- For me personally, getting a practical example of how Scrum can work in practice is very important and facilitates my job learning to be a good scrum master
- For my team, it is very important that they got an idea of how Scrum works and why it is important. It will help me when trying to motivate them to respect its principles, procedures, events and artifacts — the "overhead" it brings.

- It also worked as a quick and effective way to build some sense of team unity.

You may also comment on the game, what was good/what could be improved, any feedback on game is very welcome!

- The game was good, but it would have been interesting also if there would have been the added complexity that 1 person at each iteration would switch the team. Thus requiring the exercise of additional communication for the new member but also to give people the chance to see how different teams might work differently. Of course that might require 7 sprints for teams of 7 members for everyone to see another team which is why it might not be always feasible.

2 Project review I

This entry was written on December 5, 2015 just after our first project review. First I'll describe three of the educational observations I have made during the first third of the project. Second, I'll summarize my contributions to the project so far.

First, Scrum seems to be a very reasonable software development methodology. It is lightweight yet it provides a suprisingly robust basic structure and system to build on. While giving "top-down" some structure it manages to acknowledge the human aspects of teams and team members and to not restrict free thinking by too many rules and division of roles, responsibilities and processes. It seems to really do what it claims to do: emphasize the importance of and facilitate performance in transparency, inspection, and adaptation. Moreover, it leaves room for the varying personalities, skillsets and desires of team members.

Second, in the beginning I had to study and prepare a lot for each scrum event, but currently I feel very little stress about them. The two events that are still challenging and require more inspection are daily scrums and retrospectives, perhaps because I saw them as the least challenging early on. Eliciting meaningful but brief output from myself and our developers in daily meetings has not been easy, though we have been doing visible progress in the application of the practice.

So far we've been summarizing everybody's daily scrum output into our meeting notes file. This formality and the requirement that we don't finish until we have logged some meaningful output from each member was necessary in my opinion in order to make sure everyone gets used to really following the practice.

In the future we might stop doing that and make the scrums shorter, expecting that developers are more able to bring up noteworthy stuff without any elicitation effort. The same holds true for our retrospectives. In the future we will likely make the retrospective sessions slightly more compact expecting that team members have already thought about relevant stuff prior to the session and are more used to giving out feedback and sharing their thoughts. I plan to also add some more structure, perhaps some sort of a discussion frame (or game), so that most important areas are dealt with effectively.

My third observation is about a problem with Scrum that we have faced. In my opinion the successful application of Scrum seems to require that team members are not only

devoted, but both skilled and experienced.

According to the course material the development team should be self-organizing and independent. However, in my opinion a group of inexperienced students such as in our team cannot be expected to very effectively organize themselves quickly to immediately start yielding top results. I feel that there should preferably be at least two energetic individuals per development team to boost the pace of the group and at least two people with strong knowhow and experience in relevant areas in order for the team to be able to smoothly self organize itself, study the problem, design an architecture of the solution, and quickly start generating quality software.

Our team had limited software development experience outside the basic Aalto programming courses. Similarly, I found that there was only a limited amount of the stereotypic American MBA graduate spirit among the personalities of our team. I felt that the potential of our team members could be effectively drawn forward only by a strong leader, almost a top-down director at times, and that the team members actually wanted that kind of clear leadership.

Also, not all members had the intrinsic thirst for achieving great success in general. Several members seemed to be fine with just about passing the course. We managed to heal this divergence by organizing a few sessions where we discussed the benefits of excelling at the project, what it would take, and how realistic it would be. I shared some experiences I had had and also presented a set of slides based upon scientific research on successful teamwork presented to me on another Aalto course.

Anyhow, subsequently I have felt that I have been forced to take up roles of not only the scrum master but also the communicator (all stakeholders), the director (planning, scheduling, top-down direction), the team leader (motivator, psychological support), the architect (overall design) and the lead developer and doer (plans, artifacts, harder programming challenges).

This uneven experience and power structure unsurprisingly produced some conflict and inefficiency in that some team members often repeatedly questioned my recommendations and argumentation. Consequently, a lot of time had to be spent also on communicating why and how I saw that certain non-technical and technical things should be done in certain ways.

At the end of sprint 0 I proposed to designate the role of sprint team leader to partly alleviate this problem. Mainly to share the prementioned roles of the communicator, the director and the team leader. I proposed that Joona would take the role as I saw him as the most ambitious and driven and thus the safest candidate. I thought that he would also likely give a good example to the others, as he did.

I need not mention that I and we had to put a lot of effort during sprint 0 to study and

learn about the requirements, problem domain and potential solutions related to the PO's vision. A basic web software project would have been a lot easier since in addition to me one other student in our team had done the Aalto web software development course and some had related experience while I was practically the only one with Linux sysadmin experience. Architecturally web software projects are also somewhat simpler to start with because they almost always are built on a standard MVC framework for which there are plenty of clear tutorials and guides like those for Ruby on Rails and Django. In our case there was no one with solid software development experience that would think for us how to approach the problem. Instead we had to design everything from the ground up and find relevant third party experts to consult us. Summa summarum, I considered and still cońsider the project together with its full scale of problem areas from technical to social both really interesting and really challenging.

In the end, in my opinion, we still managed to proceed pretty effectively overall. Joona did an awesome job during sprint 1 as sprint team leader and I think the team is now a lot stronger after overcoming and discussing the many challenges this project has confronted us with so far. I might have been somewhat ineffective with my time use, but I feel that I mostly only tried my best to do what I thought was necessary as fast as I could. I had lots of other business to attend to during this fall so spending my time was no problem. However, I have to admit that I enjoyed the project, and even writing this diary, which partly explains my effort.

The following depicts my main contributions since the start of the project:

1. Acted as the team's scrum master and leader – motivator, facilitator, driver – lead doer.
2. Teached and helped the team to do larger software projects as a team (building, studying and setting up services and tools like GitHub, git-flow, Floobits, Flowdock, Agilefant, Python virtual environments, Sphinx, Python unit test framework, Markdown, ReStructuredText, LaTeX).
3. Helped the team understand the project, design the solution and develop it.
4. Prepared initial versions of the scrum artifacts.
5. Programmed the initial prototype for sprint 0 review and solved critical programming problems during sprint 1.
6. Did several other things I saw were necessary in order to make sure we excel as a team.

3 Project review II

This entry was written on March 5, 2016 just after our second project review. First I'll describe three of the educational observations I have made during the second third of the project. Second, I'll summarize my contributions to the project so far.

First of all, I wish to note that I feel that the second third of the project was very different from the first, and in several ways:

During the first third myself and the whole team got a pretty good understanding and working knowledge of our project, Scrum and the course and gained important practical experience. In the second third we were able to continue implementing adjustment ideas regarding our application of Scrum and our teamwork methods and profit from our knowledge and experience capital. This increase in experience and our sprint team leader system made the second third of the project vastly easier for me as our team's scrum master and inofficial project manager.

Clearly a major reason why the sprint was lighter for me was the fact that many of the critical and challenging tasks that I was responsible for such as connecting stakeholders, creating plans and designs and proposing and teaching teamwork methods were largely done in sufficient extent and quality during the first third.

Besides the obervation above, I wish to note the following three learning observations:

1. It is difficult to say whether daily scrums and general teamwork meetings are effective or not. In some of our earlier retros we acknowledged that not all of our daily scrums looked like out of a scrum manual. In deeper retrospect I feel that they absolutely should not look like such when the team is as inexperienced as ours was. The sometimes disorderly liveliness of some of our daily scrums was a sign of wholehearted commitment, keen interest, and of a culture of inclusiveness, acceptance, and straightforward nonhierarchical collaboration.
   Sometimes I felt that we had wasted hours and hours on discussions from which I gained very little. However, in retrospect, I think those long discussions and debates are the reason most things have gone very smooth ever since.
2. Trying to achieve quality and good results in things that are not that essential by control and micro management is clearly less fruitful than letting everyone act pretty much the way they see fit. As a leader one must prioritize the long list of little

things that should be followed and done and start by presenting the list to each team member asking which two to four are they willing to commit to doing.
It can be more fruitful in the longer term if initially some little things are neglected or get patched in a hurry by the scrum master himself but the major things progress effectively while the spirit remains high and everyone is enjoying the project.

3. One should not be too ambitious when thinking about practices that one plans to be carried out in a regular basis. For example I had the idea that we would gather the team at the end of each sprint perhaps just after the retrospective and give each and every team member feedback on his/her contributions and collaboration style. However, even just the challenge of making sure we have a clearly organized retrospective as required by Scrum was enough. The retrospectives often took the time and energy our team had to spare for it and there was not enough for any additional pondering work.
Also, our developers were not so interested in spending time on unrequired work items, perhaps rightly so.

My main contributions are listed here: During the second third of the project I

1. Acted as the team's scrum master – trying to keep up the team's working efficiency (work planning, motivation), scheduling and leading scrum events, ensuring adherence to course requirements etc.
2. Acted as lead developer for some of the most challenging parts of the software (logic related to asynchronous features, inter process communication across hosts, clusters).