

Let's Grow in Scrum Practices: Some Management Lessons Learned from a Global Software Development Project

Hang Su

Seidenberg School of CSIS
Pace University
New York, NY, USA
serenesou666@gmail.com

Abstract. This paper describes some management lessons learned from a Global Software Development project undertaken in an educational setting with the students from Senegal, India, and the US. It was the first time that the project adopted Agile Methodologies and Scrum for the development of a mobile application. The students considered this Scrum experience a rewarding learning process. The relevant suggestions for Educators and Project Managers for similar projects are provided.

Keywords: Management Lessons, Scrum, Computer Science Education, Global Software Development

1. Introduction

Pace University in New York City has been working with different institutions worldwide to integrate Global Software Development (GSD) project into the Computer Science Education and student developers write software together [1,2,3,4,5]. In 2009, Agile Methodologies and Scrum were adopted for the first time for developing a mobile application.

Inspired by some Scrum practices, the author of this paper, a graduate student in Computer Science and the only working IT professional with project management experience, naturally observed some interesting behaviours of developers and compares the difference between the plan and the result based on some project metrics, summarizes certain management lessons, and provides relevant suggestions for Educators or Project Managers in doing projects.

2. Project

2.1. Project setting

The GSD 2009 project intended to develop an educational mobile application by Java ME for a *Product Owner*. It allows 5- to 6-year-old children to try exercises in 4 subjects (math, geography, reading and writing), and to check if their answers are correct immediately. Five graduate students distributed across the US, India and Senegal participated as developers and working closely with Product Owner who was a lead instructor of the project. A professional process coach watched the team and provided weekly comments.

With regard to Scrum practices, the requirements of the product were maintained by Product Owner and prioritized in the form of User Stories in the *Product Backlog*. The development process consisted of short iteration cycles under Scrum's iterative and incremental framework called *Sprints*. During the preparation period, all the student participants were given tutorials on Scrum practices, Java ME, and RTC. A set of Scrum practices has been accorded. At the beginning of each Sprint, developers discussed and determined the goal of the Sprint with Product Owner, selected the Stories and broke them down to tasks. Each developer picked the desired tasks and took every effort to accomplish them. We had totally 3 Sprints which lasted 14 days each, following a preparation phase of 24 days. The IBM Rational Team Concert 2.0 (RTC) [6] was employed as our collaborative tool. All the statistics data of this paper were produced from RTC.

2.2. The expected vs. the actual situations

2.2.1. Product

2.2.1.1. User Stories

We have planned to finish 10, 18 and 18 User Stories in Sprint 1, 2, and 3 respectively, whereas we closed 1, 2 and 12 of them correspondingly at the end of each Sprint.

2.2.1.2. Functionality

We delivered the application that included a welcome screen, a four-subject menu, 2 questions of mathematics, and all the questions of reading and writing with relevant answers verification, and scores report. As we reflected the reasons for the difference between the plan and the result, we saw that:

As in Sprint 1, everything was new; about half of tasks focused on “learning Java ME” or “testing different features of RTC” or “reading Scrum primer” and 41 hours, or about 70% of total planned time (out of totally 59.25 hours) went to these pure learning activities. The only Story closed was “Java ME learning”.

Sprint 2 was not very productive as some developers had exams and religious activities; we retarded the goal-setting meeting, omitted some planned meetings, and did not have any of them with full attendance.

In Sprint 3, we enforced team communication by designating one Scrum Master dedicated in controlling the project process. The team dragged most of Stories from previous Sprints and really “sprinted” to the end of the project and we finally closed 14 Stories.

2.2.2.1. Team work hours

The percentages of the actually hours to the planned ones are 78.9%, 94.51% and 43.97%, for 3 Sprints respectively. The distinct gap of Sprint 3 is mostly due to the fact that some developers did not close work items upon completion, the time estimated for some tasks at the later phase was not updated, and the time spent for some other tasks was not factored in RTC. As for Sprint 2, we did not start planning work items until its second week, and it was much easier to estimate the time needed for 1-week tasks than 2-week ones. That is why we have a high percentage of 94.51%.

2.2.2.2. The Burndown Chart

The trend of burn down does not indicate an optimistic result as we did not pull down the remaining work curve. However, it should be pointed out that the chart here is solely for the reference, as it does not completely reflect the real situation as many items were not closed upon the completion and the work items did not 100% accurately log the reality.

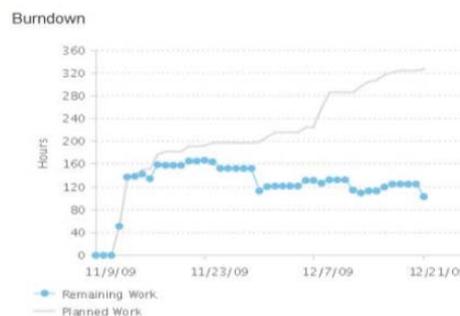


Fig. 1: The burndown chart for the entire project

2.2.2.3. Team Roles

As a learning process, the team agreed to rotate the role of Scrum Master among developers. Scrum Master mainly helps the team remove impediments and ensures the project progress. The author proactively practiced this role in Sprint 1 and learned by doing. In Sprint 2, the predefined Scrum Master did not carry out the due work and we lost the track of the progress. The planned Stories were not integrated and the team received a serious warning from Product Owner. At the end of Sprint 2, the author sensed that it was time to take over this role and hopefully that could help foster communications and ensure the product delivery. In Sprint 3, with the coordination of the lead professor and the process coach and with the agreement of the other developers, the author became Scrum Master, and spent most of time in managing the team and process so that she had to give up certain development tasks which were planned before Sprint 3. That was part of reason why the whole mobile application did not make the questions of all subjects ready at the end.

2.2.2.4. Daily Scrum Accomplish Rate

The team agreed to write Daily Scrum with quality wording – by specific and clear descriptions and post them in RTC before starting each working day. We should mark down: one’s work on the day before, the

plan for the current day and any obstacles. In case of forecasted absences, notes like “Not a working day, I will be back on <date>” should be posted.

The accomplish rates for Sprint 1 and 2 are only about 50% and if we consider Sprint 1 as a trial period, we felt embarrassed about the same rate for Sprint2: the process has been dragged along very slow, some developers have not updated Daily Scrums for 10 consecutive days; the poor wordings such as “Try to do tasks allocated to me”, or “Unstable Internet” were overused. We barely had updates in Sprint 2. Fortunately in Sprint 3, with the supervision of the new Scrum Master, the average rate increased to 58.57%. The quality was improved too: the more concrete descriptions have been recorded, e.g.: “I am going to do Task 223 that is part of User Story 79” and “I have an exam today and tomorrow, will be back next Tuesday”.

2.2.2.5. Self-organizing activities

We developers were supposed to communicate constantly, learn from each other, reflect skills and lessons learned, and push forward the development work together. There is no better practice than this to nurture ourselves, as an individual and as a team. One of the best indicators is to see if the team actively held meetings and talked to each other outside of the planned meetings led by professors. In our records, only in the final phases, have 7 self-organized online meetings and exchanges among developers been witnessed, after a supervisory role aimed at enhancing communication was added.

2.2.2.6. Source Control, Repository, Documentation Management and Maintenance

The team had planned to use RTC to log all the exchanges of messages and documents occurred, and to depend on its repository to see the real-time development progress. As the team was new to this tool, emails and online chats became our main communication media at the most of time in the preparation phase and the first half of Sprint 1. And we found that we were naturally inclined to hold the tools we were more comfortable with, and were mainly pushed to give up them and returned to RTC. In Sprint 2, we did not really progress much so the exchanges reflected in RTC remained as low as Sprint 1. With the supervision of new Scrum Master, Sprint 3 recorded obviously more documents (meeting minutes, chats, work items change logs, and associated email notifications).

3. Some management principles learnt and suggestions

With reference to the above comparison and post-project interviews, the author concluded with some important management principles, based on which she would like to provide Educators or Project Managers relevant suggestions for project practices.

3.1. Lesson 1: “Everything Starts with the Will!”

The students were expected to be highly self-motivated to take extra work for this project in addition to their original coursework. All the developers realized that, in exchange for voluntary efforts, they would gain knowledge, skills, experience and camaraderie. Everyone seemed excited on Day 1 and they agreed to commit to a set of Scrum practices. However, we have seen different cases that participants lost or lacked motivation, did not report daily progress, did not commit to planned rituals, and felt reluctant to communicate. The interviews showed that some of them preferred code writing much more than administrative reporting, and they also considered this a safe school project so they were not threatened by “breaking the rules”.

Suggestion:

Educators or Project Managers shall ask students to write down their motivation and commitments and use them to remind students--especially as their enthusiasm flags over the course of time, combined with constant psychological reassurance, along the lines of simple empowering slogans such as "You can do it!" and “Don't feel bad if you make mistakes – only those who don't work make no mistakes!”

As for the preference problem, students need to be told that principles are intended to provide a regulatory frame of reference for the project. No matter what you do, whether at work or among society, you need to follow the pertinent rules regardless of personal interest. One should seek a wider perspective, seeing the whole picture and relinquishing petty pride to comply with them, since they are necessary means to accomplish a greater purpose. Let your motivation inspire you and guide you!

3.2. Lesson 2: “Recall the 20-80 Rule!”

As this project required specific programming skills and software engineering concepts notably the Scrum with which most developers were unfamiliar, the team spent 24 days (in preparation phase) and about 70% of total planned time in Sprint 1 in learning all these new. The later practices taught us that this "overemphasize-everything" approach was unnecessary and unattainable. A handful of frequently used

features and basic operations within the repository would be enough. The famous 20-80 rule revealed its truth: 20% of tasks have priority over the remaining 80% and require 80% of the time.

The author should point out that most students, according to their feedback, had some prior experience of this rule, but apprehended it this time in a more direct, visceral, way – as a team we committed this mistake together and lost a lot of time together. We could have used the time better.

Suggestion:

Educators or Project Managers should pay attention to students' activities and stress this rule constantly. Students should be given immediate warning on their attempts to master details on everything. Some industry practice principles like “prioritize your tasks (do the right thing) then work on the effectiveness (do it right)”, or “don’t lose 100% of market for a 1% deficiency” need to be repeated. The author’s experience suggests that they can’t be emphasized enough!

3.3. Lesson 3: “Time Management: Keep Practicing...”

Most students budgeted time inefficiently, especially at the beginning of the project. They devoted too much time to tasks relatively unimportant and trivial in overall product functionality, such as “trying to learn RTC” or “looking for the picture for welcome screen”.

All students expressed their willingness in improving their skills in time management. Interestingly, most students said that they knew some principles of good time management, such as “do the most important and urgent tasks first”, and good habits like “finishing today's tasks today”. These are all important within Scrum practices; however, they were adjudged “difficult to put into action” and dismissed with “words are 1000 times easier than actions”. But they admitted that regulated by Scrum rituals and the obligatory adherence to them, this project provided a good chance to understand all of these in a more explicit and direct manner, as they saw results of bad time management and bad habits right away.

Suggestion:

Educators or Project Managers should convey lessons like “we cannot afford to lose clients when it comes to product delivery”, and “victory goes to the swift”. Students’ allocation of time should be watched, and they should be reminded to concentrate on the tasks that matter most, that really carry out the most functions of the application based on the Sprint plan.

3.4. Lesson 4: “Be Cool...Keep a Constant Working Pace!”

Some developers were so very passionate about technical skills, mainly programming that they attempted to implement Stories extraneous to the plan without reporting it, and ended up implementing less than expected due to time constraint and the difficulties of integration. Some of them burned out through all-nighters in order to finish the tasks that should be evenly distributed in 10 days. All these “outrage” behaviours – common among students – resulted in the criticisms of the Product Owner. Staying within the boundaries and complying with rules save us, in this sense. Working at a constant pace keeps us healthy, the team healthy and protect ourselves in front of customers: who would appreciate getting shocked at the sight of nothing in the repository just at the closing time of one Sprint?

Suggestion:

Educators or Project Managers should emphasize the concept of agility, which is a balanced agile speed over the whole production phase with no fluctuation, not an absolute unwavering speed at any time. A supervisory role is needed.

3.5. Lesson 5: “Ends Justify the Means!”

In delivering a small application under Agile Methodologies, we firstly need to make sure the software works. In the practices of refactory within Java ME, some developers scrutinized the coding efficiency and aesthetics too much. Nothing wrong about that, as we were all learning new techniques but we recognized more deeply that even though coding efficiency and aesthetics can be considered as benchmarks, delivering workable software must always take precedence. And as we also piloted some light peer reviews, we all agreed that refactory was about finding the bug, not about the coding style.

Suggestion:

Given this particular mobile application context, Educators or Project Managers should emphasize the concept of “software must work, computer must compute, and function over all!” The clients are the God, and only look at the ends. We need to focus on an application that works, and there lies our sine qua con.

3.6. Lesson 6: “The maximum Water Volume of One Bucket Depends on its Shortest Plank.”

If this bucket refers to one person, then his weakness is his shortest plank. Scrum practices let everyone

get exposed to different aspects of a development project: techniques, management, communication and teamwork; we gain better understanding on our own preference, strength and weakness.

If the bucket refers to one team, then the shortest plank is the weakest member, whose performance mostly decides the team's overall level. Scrum rituals require us to do a lot of communication, which just help us comprehend each other and have a good understanding of everyone's strength and weakness. One bucket can hold a certain volume of water, when every plank contributes; whereas one plank can't hold any water; just as one hero-developer cannot triumph. Teamwork counts when we work in a team, each one of us succeeding only when the team as a whole succeeds.

Scrum practices, in some ways, provide us a good opportunity to examine our shortest plank and reinforce our understanding of teamwork. One explicit example is Daily Scrum: regardless of personal willingness, developers are exposed to the whole team, which, as a psychological effect, forces everyone to be more serious, to think in the stature of the team rather than the individual. On the other hand, from what others do, we can constantly reflect on ourselves: if I was in her position, would I do the same? Anything I could improve? When?

Suggestion:

With respect to Scrum context, Educators or Project Managers should illustrate and emphasize the notion of teamwork, along with the willingness to shoulder responsibility. A dedicated supervisory role shall be added, as it will most likely ensure overall communication.

A vivid example with a bucket consisting of different-height planks might be employed as a prop in the due course as the project moves on, students can be asked to think about this metaphor and reflect on themselves, and to discuss with team-members on the bucket analogy to the project they participate in.

4. Conclusion

We thanked Scrum's simple but strict practices, which forced us to go through a cycle of speculation, collaboration, learning by doing and adjusting, filled with opportunities for self-reflection and self-improvement. We felt grateful to learn all the important management principles, which are as important as some other "real" techniques: Java ME, refactory, RTC, and some software engineering concepts.

However, to use Scrum in a more efficient way, adding a strict supervision role is indispensable. It not only gives students a hand when necessary, but also assures the accuracy of project data. Some students acknowledged a better team performance in the later phase by the new Scrum Master and stated that a dedicated supervision role is necessary.

5. Acknowledgements

The author would like to thank Dr. Christelle Scharff, Dr. Olly Gotel, Prof. Vidya Kulkarni, Mr. David Soler and all the students involved for the support in writing this paper.

6. References

- [1] Gotel, O., Kulkarni, V., Say, M., Scharff, C. and Sunetnanta, T. "A Global and Competition-based Model for Fostering Technical and Soft Skills in Software Engineering Education". *Proc. of the Conference on Software Engineering Education and Training (CSEET 2009)*, Hyderabad, India, February 17 - 19, 2009.
- [2] Gotel, O., Kulkarni, V., Neak, L. and Scharff, C. "Working Across Borders: Overcoming Culturally-Based Technology Challenges in Student Global Software Development". *Proc. 21st Conference on Software Engineering Education and Training (CSEE&T 2008)*, Charleston, South Carolina, US, April 14-17, 2008.
- [3] Gotel, O., Kulkarni, V., Neak, L., Scharff, C. and Seng, S. "Introducing Global Supply Chains into Software Engineering Education". *Proc. 1st Intl. Conference on Software Engineering Approaches For Offshore and Outsourced Development (SEAFOOD 2007)*, Zurich, Switzerland, February 5-6, 2007.
- [4] Gotel, O., Scharff, C. and Seng, S. "Preparing Computer Science Students for Global Software Development". *Proc. 36th IEEE Annual Frontiers in Education Conference. Borders: International, Social and Cultural (FIE 2006)*, San Diego, California, US, October 28-31, 2006.
- [5] Gotel, O., Scharff, C. and Seng, S. "Incubating the Next Generation of Offshore Outsourcing Entrepreneurs". *Proc. Symposium on Information Technology and Entrepreneurship (ITE 2005)*, Oklahoma City, US, April 19-20, 2005.
- [6] IBM Jazz. See: <http://www.jazz.net> (accessed April 2010).