My Experience in Global Software Engineering using Scrum Practices

Jyoti Sheoran
Department of Computer Science
University of Victoria
Victoria, BC, Canada
Sheoranjs24@gmail.com

***Abstract*— With the increase in usage of distributed scrum in industry, it is important for students to gain knowledge of best practices in global software engineering. The author describes her journey throughout the course - challenges faced and lessons learnt.**

***Index Terms*—Global software engineering, distributed scrum, agile practices.**

I. Introduction

Global Software Development (GSD) [1] has become the norm and so is agile software development methodology [2]. There is an increasing trend to combine agile software practices with GSD practices [3]. It is therefore very important for students to learn and gain hands-on experience with GSD using agile practices.

I am a graduate student at University of Victoria. I did my undergrad in Computer Science from Maharshi Dayanand University, India. I also have 21 months of experience working in a GSD team in Amdocs Development Centre India. My team was divided in 4 different locations – 2 sites in India (Gurgaon and Pune), one in Israel and one in Dallas, US. When I joined the company, I was given cross-cultural training. But I was thrown in the pool to fight and learn GSD practices myself. I learned that it takes time to develop trust in distributed team, and only way to achieve it is by delivering quality work. Other things that I learned - frequent communication with other sites, understanding cultural differences and time-zone differences. My main area of work was developing change requests based on High Level Design documents. I was not aware of best practices and challenges in dealing with customer, gathering requirements, project planning, etc. in global teams.

With the aim to learn software life cycle and best practices in GSD using scrum practices, I took the GSE course in Spring 2014 at University of Victoria, instructed by Dr. Daniela Damian. This paper describes the challenges and lessons learned by the author during this course. The paper is structured as follows: Section II presents current state-of-art in GSE using scrum practices, Section III describes course structure, Section IV presents initial hurdles faced by the author, Section V discusses the experiences of the author in this course, Section VI describes the challenges that she was unable to mitigate and Section VII presents the lessons learnt. Finally, Section VII presents the conclusion.

II. Literature Review

Globalization has become a common and significant practice in software and IT industry. Various factors that accelerated the adoption of GSD include low-cost qualified workers, involvement of local population and cross-site modularization of development work [4]. However, GSD also has its challenges. The main challenges in GSD are cultural differences, communication, coordination problems, knowledge management and concurrent engineering principles [5]. Bikram Sengupta talks about people factors in Globally Distributed Projects (GDP) [6]. He mentions that for global software projects, industry prefers to hire people with good communication skills and team spirit. In GDP, it is important to understand cultural differences and build trust among different sites.

Project manager also plays an important role by giving freedom of decision-making and leadership in local teams and promoting equality among teams.

The case study done by Herbsleb et al. [1] describes that in early days of GSD, there was lack of definitive practices. Most of the practices emerged by trial and error. The best practices in GSD are – synchronous milestones, frequent delivery, same iteration cycles in all sites, peer-to-peer links, weekly meetings, maintenance of progress report and use of different communication tools such as email, instant messaging, video-conferencing, etc.

One of the biggest challenges in GSD teams is cross-cultural issue. The best practices to manage cross-cultural issues are – use of culturally neutral software, agreed process and tools, establishment of bridging staff, negotiation of cultural and time differences by giving incentives to employees and providing cultural training [7].

Another big challenge is knowledge transfer among stakeholders from different cultures. The best practices to establish a shared understanding between stakeholders is – open communication, regular monitoring, peer-to-peer links and establishing cultural liaisons [8].

Different time-zones is a disadvantage in GSD, but some companies tried to convert it into an advantage by

implementing practices like Follow-the-sun and 24-by-7 work However, such GSD projects took longer to finish than collocated ones. Follow-the-sun methodology is difficult to realize in software projects because of challenges in perfect hand-off and coordination [9].

Mockus et al. describe different ways to divide work in globally distributed teams [10]. The work should be divided in such a way that it minimizes interdependency and communication among different sites. Various ways to divide work in software teams are – division by functionality, localization, development stage and maintenance stage.

Lanubile et al. [11] gives an overview of various collaboration tools for GSD. There are many collaborative Development environments (CDE) (e.g. GitHub, SourceForge, etc.), but none of them supports all collaboration activities in GSE. GSD teams use multiple tools to perform collaborative activities.

Agile practices (Scrum [12] and XP [13]) support iterative and incremental software development, where requirements and solutions evolve through collaboration between cross-functional collocated teams. Agile practices have many advantages that mitigate the challenges of GSD [14] [15] [16]. For example, communication among stakeholders is a challenge in GSD and scrum promotes the practice of frequent communication using daily scrums and scrum-of-scrum meetings.

Maria Paasivaara et al. [17] presents a case study of using scrum in a Globally Distributed Project. Along with scrum and GSD practices, the company was also using some non-scrum practices – nightly builds, automated testing, team rooms, etc. Using distributed scrum in global teams has its own challenges, such as synchronous communication, lack of all-in-one collaborative tools and management in large teams [18]. Hence, application of only agile practices to GSD projects is not sufficient and one must use other practices along with scrum and GSD practices.

III. Course Structure

This course was collaboration between University of Victoria, Canada and Aalto University, Finland. There were 4 teams each consisting of 4-6 members with different technical knowledge and cultural background. We had a real product owner and a chief architect from Finland.

A. Project Description

The project was backlog management software for mobile devices. The product owner already had desktop website for backlog management. Our project was to develop the front-end for mobile phones that will use same backend as desktop website.

B. Project Develoment Process

The project was four month long with first two sprints of 1 week long and next 4 Sprints of 2 weeks duration. In the first Sprint, Canadian students were divided into 4 local teams. For the second sprint, each of the four teams had 2 extra members from Finland. In next two sprints (3rd and 4th), two teams were collocated (one in Canada and other in Finland) and other teams were kept same (i.e. members from both sites). In the last two sprints (5th and 6th), collocated teams changed back to same as in Sprint 2, and other two teams became collocated. Each team had a scrum-master, which kept on changing every sprint. Scrum practices were followed – 2-week sprints, sprint demo, sprint planning, twice a week scrum meetings, weekly scrum-of-scrum, sprint retrospectives and final project presentation.

C. Project Development Tools

The technologies used in the project are – HTML, CSS, JavaScript, AngularJS, REST API, JSON and XML. The tools used in the project were InteliJ IDE, Eclipse, Sublime, GitHub, Maven, Flowdock, Google-hangout and Google docs.

D. Project Initiation

Course began with a Kick-off party. The product owner, Finnish professor and two Finnish students visited University of Victoria. They helped the Canadian students to setup the development environment and gave project related technical tutorial.

IV. Initial hurdles

Though I learned agile methodologies (XP) in my undergrad courses, I didn't have any practical experience of using agile methodology. My industry experience was only with Waterfall model. Below is a list of things that involved a lot of learning from my end:

1. Scrum practices
2. GSD using Scrum practices
3. Relationship management with Product owner
4. Global requirement engineering
5. Global project planning and story estimation
6. New technologies such as AngularJS, REST API, web-based app development for mobile
7. New tools – InteliJ IDE, GitHub, Maven and Flowdock

V. Experiences

The experience I gained from this course is invaluable and will certainly help me in the industry. Below is a summary of my experiences during various phases of the project:

A. Sprint Planning

Each team was free to choose any story in the Sprint Planning. This gave us both the freedom to choose and the responsibility to evenly divide stories among the teams. In the first Sprint, we choose random stories that we found interesting and that can be finished in the Sprint. In the 2nd Sprint, our plan was to pick stories related to previous sprint stories. However, I was amazed that some teams jumped into stories that were either deferred by other teams or closely related to other team's stories. For example, one team, say A choose the story that was similar to story finished by team B. I hinted team A that team B has already done work on same thing. But they still took that story. At the end of Sprint, team A deferred the same story and

team B took it. Story choosing got better with time among other teams.

In the final Sprint (6th), there was one story – 'developer guide'. None of the four teams raise hands to take this story. There was complete silence for 2 minutes. Product owner asked the teams to volunteer for the story. Finnish team said that native English speakers should take this story and got their hands-off. Canadian local team said that they already have too much work and they didn't want to defer stories. Now the choice was between two teams who had members from both sites. None of the teams came forward to take the story. Finally product owner had to toss a coin. I wonder what happens in industry if none of teams is willing to pick a story!

Other important activity in Sprint Planning is story estimation. In the initial Sprints, my estimations were mostly wrong, as I was new to the project. But with time and experience, my story estimation got better. One myth that I had in the starting was – story point is related to the difficulty level of the story. But later I realized that story point is related to the effort required (in hours) for a story. This myth also contributed to wrong story estimations in the initial sprints.

B. Sprint Retrospectives

Thanks to one team member who had experience in Scrum practices, the Sprint retrospectives were very effective in our team. Each team member in our team had to write 4 points in a shared spreadsheet, which can be positives or negatives of the last sprint. We discuss about these points and select 4 most important points by voting that we should continue or improve in the next Sprint. Usage of white board during the discussion was very effective when our team was collocated.

C. Sprint Demos

In the initial sprints, I felt that Sprint Demos are merely a formality, as we already gave story demos to product owner and got his approval. But after constant reminder from our instructor, I realized that Sprint Demo is not just for the product owner, but it is to create a shared understanding of the product across all project teams. Quality of my sprint demo increased as the course progressed.

D. Scrum Meetings

I didn't have any trouble in scrum meetings, as I had experience of daily meetings in my previous company. Presence of an experienced team member was also a huge bonus for our team. We structured our scrum meetings on three questions: 1) What I have done since our last meeting? 2) What I am planning to do next? 3) Do I have any blocks or do I need some help?

I also participated in scrum-of-scrums held every Sunday. Only one member from each team was required to join the scrum-of-scrums. But it didn't work as indented, largely because the people who joined the scrum-of-scrums didn't have enough knowledge about the blocks and issues of their fellow teammates. One of our team member suggested that all of us should join the scrum-of-scrums with only the scrum-master doing most of the taking and other team members can jump in the talk if required. Scrum-of-scrums helped me in developing a shared understanding of project issues and blocks of other teams. It was also a good place to talk about problems affecting multiple teams.

E. Communicaiton and coordination

I come from a strict hierarchy based culture. It took me some time to realize the idea of equality and importance of taking initiative promoted by scrum. I took initiative in joining Community of Practice and scrum-of-scrums.

Initially, I contacted the product owner and architect only if I had completed the work or if there was some serious problem. But later I realized that we should ask as many queries as required in the beginning rather than in later stages. My communication with product owner and architect became more efficient with time.

The coordination within my team was good. We divided our stories in such a way to have minimum coordination requirement between local and remote team members. Mostly, our stories were independent from each other. Finnish team members took stories that required frequent communication with the product owner and architect, as they were collocated.

There was little to no interaction between me and other teams. It wasn't much of a problem in the initial Sprints, as I didn't have any story that required coordination with other teams. However, it became a problem in latter sprints as our team members got split into different teams who were working on an incomplete story. I had to resort to frequent communication and pair programming to finish that story.

F. Cultural Differences

We were given an introduction of both Canadian and Finnish culture before the start of the project. I also made extra effort to read more about both cultures. Even after reading about these cultures, sometimes it was difficult for me to acknowledge it. There is one particular aspect that I found difficult to deal with. I come from a very straightforward culture in India. Finnish are also straightforward and hence I jelled very well with Finnish team members. On the other hand, Canadian team members were very polite even if there was a big issue. This politeness made me difficult to understand the severity of problems. The effect of this can be clearly seen in our team's progress. Our team performed best with almost 100% story completion when we had a mix of Canadian and Finnish team members. However, our team performed worst and deferred stories when we had only Canadian team members. Though I understand Canadian culture very well, but it was still a challenge for me.

G. Requirement Engineering

Requirement Engineering is challenging when the product owner is at remote location. In the first Sprint, I struggled in understanding the requirements. There was no story description. The product owner was not very clear about what he wanted and he kept changing the requirement when I demoed the stories to him. Hence in subsequent sprints, I contacted the product owner in the beginning to clarify the

story and its requirements. I made User Interface diagrams and got his approval before I started to code. When I demoed the story to him, he was mostly happy as I did the work we agreed on. For some stories he did say that he wants something else, but I reminded him that this is what he asked me to do and if he wants something else, then he has to modify the story and it will be implemented in the next sprint.

H. Relation with PO

In my previous work, I never had a chance of direct interaction with the product owner. Product owner was typical in the sense that he wanted to implement more features and didn't pay much attention to the quality of the product or its scalability. After two sprints, we realized that working in this manner will have long term repercussions and we started to negotiate with the product owner to let us work on testing framework and UI design. He agreed but didn't create any story for it. As a result finishing stories was always a higher priority and testing kept on being neglected. So, in the fourth sprint, we asked for dedicated hours and made stories for testing and UI design.

I. Technical knowledge

I had experience working in an industry where management simply throws new technologies on us and asks us to read documents and start working on it. I was new to technologies that we used in this project. But this didn't scare me and I managed well to learn these new technologies and work with them. The first Sprint was a high learning time for me. But after this, things became easy.

One difficulty that I had in the project was the lack of documentation for REST API. Even the code was not commented. We needed REST API for communicating with the backend server and its knowledge was crucial to complete the stories. However, without any technical documentation and comments it was very challenging to gain knowledge about the API. In the first Sprint, our team carefully selected those stories that require minimum interaction with the API. But we couldn't save ourselves in second sprint. It took me one whole sprint to get familiar with the API, but it was totally worth it. I can proudly say that I can work on legacy code with no documentation.

J. Knowledge management and documentation

Because we were not given any documentation, most of us didn't feel the need to document our own code. But some experienced members in our project constantly reminded us to properly document our work - write proper comments and add description in stories and GitHub pull requests. As the code base kept growing, the importance of documentation couldn't be ignored. I tried to document as much work as I could. Properly documenting my work helped others understand my code.

VI. CHALLENGES

This section lists the major challenges that we were not able to solve in this project.

A. User Interface Design

We had no UI framework or UI design guideline in the beginning. This became more troublesome as we were aiming two different mobile platforms – Android and iOS. While Android is lenient on the UI, Apple has very strict UI design guidelines for iOS. As a result, in almost every sprint, we kept on changing the UI that led to breaking of other features.

B. Product Testing

Product owner was reluctant about testing. As a result no testing framework was setup in the starting of the project. Later when we tried to add a testing framework for unit testing, but it was too late, as this would have required re-writing the whole code to make it unit testable. This was a bad mistake. Product testing is crucial for quality and must be finalized at the start of the project. However, we did resort to exploratory testing. We made lots of manual test cases and assigned it to all teams. Product owner also encouraged us to use mobile app for our backlog management and report bugs.

C. Documentation

Apart from commenting our code and adding description in stories, commits and pull-requests, there was no provision of maintaining a formal documentation. Scrum doesn't say anything about the documentation. And in lieu of following scrum practices, we completely ignored the importance of documentation.

D. Roles

Yet another challenge we faced was to handle those team members who were not following the scrum practices. As everyone in the team was equal and there was no manager or team leader, there was a question of who should make sure that scrum practices are properly followed. No one wanted to be the one to point out someone.

E. Project management

Scrum practices are very effective for GSD projects. But these practices are for software development. Scrum practices do not say anything about project management. We ignored this fact and did not use any other project management tool apart from backlog management tool.

F. Repetitive work

There was a lot of duplication of effort due to various reasons like different teams picked similar stories, change in UI, change in requirements and implementation of new features that broke old functionality. There was no way to avoid this repetitive work.

G. Issue Tracking

We used GitHub for software version control. GitHub serves well as a distributed version control, but its issue-tracking tool is not sufficient. We were using Agilefant for backlog management. And people were reporting issue on different channels - Agilefant, GitHub and Flowdock. Most of the issues raised on GitHub were not updated.

H. Collaborative Development Environment

There was lack of comprehensive CDE. We used GitHub for version control, Agilefant for backlog management, InteliJ as IDE and Flowdock for communication. But there was no integration between these tools (except the GitHub activity feed on Flowdock).

VII. LESSONS LEARNT

Below is a highlight of the lessons that I learnt from this course: -

1. Best practices in GSD using scrum
2. Software life cycle in distributed scrum
3. Global requirement engineering
4. Project planning and work estimation
5. Division of work in GSD projects
6. Negotiation with the Product Owner
7. Effective communication, coordination and teamwork
8. Collaborative tools for GSD projects
9. Working on legacy code
10. Leading technologies for web-based mobile development

VIII. CONCLUSION

The involvement of a real product owner with real project increased the sincerity of the students. This course was a good emulation of real job where product owner tries to give more work and pushes for more functionality in the product. And developers are thrown to new technology with no proper training.

We faced a lot of challenges during this course. But facing these challenges was the beauty of this course, as these challenges taught us the real problems in global software development using scrum practices and how we can mitigate them. Simply reading a book or taking a theoretical course cannot make us understand the real challenges and best practices of distributed scrum. We faced challenges, applied best practices and witnessed improvements.

This course required a lot of effort from the students. It was difficult for the students to manage time and coordinate well, as they had other courses as well. This course should have more credits than other courses.

I am very grateful to Dr. Daniela Damian and Dr. Casper Lassenius for bringing a real project to students. It would have been a challenge to arrange such a course and grade it.

REFERENCES

[1] J. Herbsleb and D. Moitra, “Global software development,” Software, IEEE, vol. 18, no. 2, pp. 16–20, Mar/Apr 2001.

[2] VersionOne, “Fifth annual ”state of agile development” survey,” July 2010, referenced: 16.01.2012. [Online]. Available: http://www.versionone.com/pdf/2010 State of Agile Development Survey Results.pdf

[3] M. Paasivaara and C. Lassenius, “Collaboration practices in global inter-organizational software development projects,” Software Process Improvement and Practice, vol. 8, no. 4, pp. 183–199, 2003.

[4] Conchúir, Eoin Ó., et al. "Global software development: where are the benefits?." Communications of the ACM 52.8 (2009): 127-131.

[5] G. Olson and J. Olson, “Distance Matters,” Human-Computer Interaction, vol. 15, no. 2, pp. 139–178, Sep. 2000.

[6] Ebert, C. (2011) Practice: People Factors in Globally Distributed Projects, in Global Software and IT: A Guide to Distributed Development, Projects, and Outsourcing, John Wiley & Sons, Inc., Hoboken, NJ, USA. doi: 10.1002/9781118135105.ch30

[7] S. Krishna, Sundeep Sahay, and Geoff Walsham. 2004. Managing cross-cultural issues in global software outsourcing. Commun. ACM 47, 4 (April 2004), 62-66. DOI=10.1145/975817.975818

[8] Damian, Daniela. "Stakeholders in global requirements engineering: Lessons learned from practice." Software, IEEE 24.2 (2007): 21-27.

[9] Carmel, E. and J. A. Espinosa (2012). I'm Working While They're Sleeping: Time Zone Separation Challenges and Solutions, Nedder Stream Press.

[10] Mockus, Audris, and David M. Weiss. "Globalization by chunking: a quantitative approach." *Software, IEEE* 18.2 (2001): 30-37.

[11] Lanubile, Filippo, et al. "Collaboration tools for global software engineering." IEEE software 27.2 (2010): 52-55.

[12] K. Schwaber and M. Beedle, Agile Software Development with Scrum. New York, United States: Prentice Hall, 2001.

[13] K. Beck and C. Andres, Extreme Programming Explained, 2nd ed. Addison-Wesley Longman, 2004.

[14] Holmström, Helena, et al. "Agile practices reduce distance in global software development." Information Systems Management 23.3 (2006): 7-18.

[15] Berczuk, Steve. "Back to basics: The role of agile principles in success with an distributed scrum team." Agile Conference (AGILE), 2007. IEEE, 2007.

[16] Sutherland, Jeff, et al. "Distributed scrum: Agile project management with outsourced development teams." System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on. IEEE, 2007

[17] Paasivaara, M., Durasiewicz, S. and Lassenius, C. (2008), Using scrum in a globally distributed project: a case study. Softw. Process: Improve. Pract., 13: 527–544. doi: 10.1002/spip.402

[18] Hossain, Emam, Muhammad Ali Babar, and Hye-young Paik. "Using scrum in global software development: a systematic literature review." Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on. Ieee, 2009.