Distributed Scrum for Large-Scale and Mission-Critical Projects

Juyun Cho
Utah State University

Follow this and additional works at: http://aisel.aisnet.org/amcis2007

Recommended Citation
http://aisel.aisnet.org/amcis2007/235

This material is brought to you by the Americas Conference on Information Systems (AMCIS) at AIS Electronic Library (AISeL). It has been accepted for inclusion in AMCIS 2007 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
DISTRIBUTED SCRUM FOR LARGE-SCALE AND MISSION-CRITICAL PROJECTS

Juyun Cho
Utah State University
jcho@cc.usu.edu

ABSTRACT

Distributed software development (DSD) is becoming popular due to the potential of delivering high-quality software products in a timely and economical manner. Recently, practitioners started to apply agile software development methods to their DSD environment. However, there are not many empirical researches that have reported whether the agile methods are really viable for distributed software development. This paper illustrates the issues of the Scrum software development method in a geographically distributed software development team setting as it applies to large-scale and mission-critical projects. Through an in-depth case study, we examine how communication, coordination, and control issues are managed and what technologies are expected to mitigate the difficulties arising in distributed Scrum (DScrum). Finally, we suggest several ideas to overcome the issues discovered in this study.

Keywords
Agile software development methods, extreme programming, distributed Scrum, distributed software development,

INTRODUCTION

Lightweight agile software development methods emerged in mid 1990s. Since then, the agile methods have gradually replaced predominant conventional heavyweight, document-driven software development methods. This phenomenon is mainly due to the potential of the agile methods to provide a higher customer satisfaction, lower bug rates, a shorter development cycle, and a quicker adaptation of volatile business requirements (Bohem and Turner, 2004). One of frequently used agile software development methods is eXtreme Programming (XP). XP was initially recommended for small-scale projects (Beck, 2000), and mainly applied to simple projects even though it occasionally applied to large-scale projects (Bowers et al., 2002; Cao et al., 2004; Lindvall et al., 2004; Lippert et al., 2003; Little et al., 2004). However, there has been limited evidence that XP is pertinent to large-scale projects. This problem leads to a real need to have an alternative agile approach for large-scale projects. One possibility is Scrum, which is a team-based light weight software development method. It is claimed to be suitable for all project sizes (Schwaber and Beedle, 2002). However, very little study has shown the viability of Scrum when it is used for large-scale and mission-critical projects. Further, the majority of research on the agile methods was conducted in environments where all of the developers were in the same geographic location.

In reality, however, as the world is getting flatter (Friedman, 2005) in the sense of globalization in information technology the more global companies are getting involved in distributed software development (DSD). Also, DSD is becoming the standard norm in many global organizations. Previous research reports that DSD can provide many potential benefits such as the ability to deliver high-quality software products in a timely and economical manner, the opportunity to hire more skillful and knowledgeable software developers, the sales advantages of having proximity to the market, and the possibility of using different time zones to create ‘follow-the-sun’ development (Damian, 2002; Herbsleb and Moitra, 2001; Sahay, 2003). In addition to these potential benefits, some of the global organizations started to apply agile software development methods to their DSD settings to gain more flexibility and agility. For example, two global companies, Hewlett Packard and Intel Corporation, have been using agile methods among their geographically scattered developers for about three years.
Cho Distributed Scrum (Holmstrom et al., 2006). Although there is little doubt that DSD can provide great benefits, the literature shows that DSD is confronted with many challenges to be overcome to take advantage of all the potential benefits. Communication, coordination, control, and trust issues are reported as the most common challenges in temporally, geographically, and socio-culturally different DSD environments (Ghosh, et al., 2004; Holmstrom et al., 2006; Ramesh, et al., 2006).

As interest in distributed software development using the agile methods grows in the software industry, it becomes very important to study the viability of distributed agile methods in managing virtual teams that are scattered geographically. In particular, it is worthwhile to study the distributed Scrum software development method which is claimed to be suitable for large-scale projects.

The rest of the paper is structured as follows. An overview of research background and research questions are given to explain where the study was conducted and what the main goals of this paper are. Then research methodology was described to present how the data were collected and analyzed. Next, the research findings are summarized and then several suggestions are given to overcome issues found through this study.

**RESEARCH BACKGROUND & QUESTIONS**

The company in the current case study has been providing mission-critical public safety software to police departments, fire departments, 911 dispatch centers, sheriff’s offices, and airport authorities since 1978. Due to the nature of software related to the public safety, a small glitch in the company’s software can cause severe disasters or casualties. As of this paper, nearly 600 agencies and over 30,000 public safety officials nationwide in the United States use the company’s software which includes records management system, computer-aided dispatch system, fire management system and jail management system. Among them, the jail management system was recently re-designed and developed under the Scrum method with five to eight software engineers for more than two years. The jail management system was mostly created using the C sharp (C#) computer programming language on the Microsoft .Net platform for front-end graphical user interfaces and C UNIX programming language for the management of a UNIX side database system. The jail system consists of more than millions of code lines. The company has a total of 30 software developers, product managers, and Quality Assurance (QA) personnel in the software development division and has been using Scrum since April, 2005. About four months after the company decided to use the Scrum method, the main part of the company moved to a different city which is about 100 miles away from the other site. Since then, the company has kept two work places and the software development team has been divided between two locations. When this happened, the author wondered whether the Scrum method would be viable for a company that deals with large-scale and mission-critical projects, and that has software developers scattered in two different geographic locations.

The main focus of the study is to examine the viability of the Scrum method in the distributed software development settings. In addition, the author tried to find the solutions for issues and problems of distributed Scrum as it applies to large-scale and mission-critical projects. The following research questions were probed through the study.

1. How do elements of Scrum work under the distributed software development settings?
2. What are the key issues of distributed Scrum for large-scale projects?
3. What technologies are appropriate to mitigate the communication, control, and coordination problems among virtual teams?

**RESEARCH METHODOLOGY**

As a first step, the author believes an in-depth case study would help to understand the issues and problems of distributed Scrum. In addition, a case study is known to be a well-suited method for capturing the knowledge of practitioners (Benbasat et al., 1987). Three types of data were gathered to triangulate findings and enhance trustworthiness (Glesne, 2006). First, while the author worked at the company as a software engineer for about two years, he participated in and observed many Scrum activities and collected data from co-software engineers through informal conversations. Second, an email survey was conducted among software developers and QA personnel. Finally, a formal face-to-face interview was conducted with a vice president of a software development division, with a lead software engineer, and with two product managers who have been analyzing and planning projects as a product manager. All participants have been using Scrum for more than two years. All of the interviews were audio-taped, transcribed, and later coded. In the process of data analysis, grounded theory (Glesne, 2006; Gall et al., 2003) was used to derive constructs from the immediate raw data. Some of questions asked during the interview included: 1) What was the most difficult problem that you have had in the distributed team environment? 2) What were the issues in the distributed team environment and how were these issues resolved? 3) How often did you have a meeting using telephone, video conferencing media, or other multimedia tools? 4) Did you have any communication problem with other team members at the other site? If you did, how did it affect your job performance? 5) Which working
environment do you prefer a co-located environment or a distributed environment and why? and 6) What did you learn, and what advice do you have for others?

KEY ISSUES OF DISTRIBUTED SCRUM

The Scrum method is mainly utilized for managing and tracking software development (Schwaber et al., 2002) using structured meetings such as the daily Scrum meeting, the daily Scrum of Scrums meeting, the sprint planning meeting, and the sprint review meeting. These meetings are key components of the Scrum method and they should be adjusted to the distributed working milieu. In this section, the author discusses how the Scrum method was tailored under the distributed environment and what issues and problems arose. In addition, the author describes what technologies were used to mitigate the issues and to resolve the problems. Through an in-depth case study, the author found information and knowledge-sharing issues were the most important issues in the company due to its geographically distributed working environment. The author also found that coordination, communication, control, training, and trust and confidence issues hinder developers from being efficient.

Information and Knowledge-Sharing Issues

When the main part of the company moved to a different location, the company went through major changes. One of them was a change in staff. According to Mike, vice president of the software development division, “There was a change when we moved to a different location with new staff and I think we lost about 15% of our staff, including software developers.” To fill the empty positions in software development, the company hired several new software developers. These newly hired people created more bugs as Mike said “There are a lot more bugs now because people are not always knowledgeable about the software that they are working on.” It is understandable that brand new employees created more bugs as Mike stated, …often they change some code, something like a radio log which is a very complex piece of code and slight changes to that have a lot of impacts in lots of areas. It takes a time to be familiar with our complex systems.

Because of the complexity of the company’s software system, it is important to have well-structured information and knowledge-sharing systems between experienced software developers and brand new software developers. In particular, it is more critical to have such a knowledge sharing system if new software developers in one location need some of the expertise of software developers in another location. One developer mentioned in the survey question “it is a little difficult to share information and explain ideas over those large distances”.

Another way to get around this problem might be placing new developers with senior developers within the same Scrum team in one location. But this solution is not always feasible for the company because of the complexity of software systems and the situation that developers often need expertise from other developers in different Scrum team and in a different location. So, to facilitate the knowledge and information-sharing, the company has been using a web-based Wiki program (http://www.wiki.org) which enables developers to add and edit items that might be critical to other developers. For example, the section called “gotcha” in the company’s Wiki includes the most frequent mistakes that developers can make in many different parts of company’s software system. Due to the easy access to the Wiki program, the information stored on the Wiki database mitigated the problems between the two sites.

In the agile methods, the code itself is regarded as all the documents that developers need. However, it is apparent that zero documents are not always right way for large-scale and complex projects, especially, for distributed Scrum environment. Also, the amount of documents should be decided based on the context of development environments though Parnas (2006) suggests a wordy document and Simon (2006) suggests no more than two pages long document.

Coordination Issues

When the company first adopted the Scrum method, five Scrum teams were organized and each team had a QA person who was designated to test the code created by his/her own team. But recently, the company pulled out a QA person from each Scrum team and created a QA personnel-only Scrum team so that a QA person can test the code generated by any Scrum teams. Because the QA Scrum team is located in one place, it sometimes causes coordination problems between remote site developers and QA personnel. For example, when developers in a remote site pass the code over to QA personnel through a concurrent version control (CVS) system, QA personnel may not know which part of code is affected by the changes that developers made. As Mike said,

The development was kind of passed over to QA and they took care of testing. But the problem with that is QA may not know what particular area could be affected. So, they can not test the area which might be broken by the changes made by developers.
This problem might be solved if developers take some parts of QA’s testing or show QA personnel the potential areas that might be affected. Another coordination issue is when new developers of the Scrum team are hired in one location developers in the other location do not try to get to know new developers. One developer stated in the survey,

When new members of the team were hired at other site, it was easier for team members here to not really put any effort in getting to know that person and learning how best to work with them. This problem was especially bad when the new members had some personality characteristics that made them a little annoying or perhaps difficult to develop a desire to want to work with them.

This issue mainly resulted from the way of the Scrum team is divided in two different sites. Other coordination problems arose when developers and support staff discuss how to divide and assign tasks between sites in the sprint planning meeting and how to track bugs reported by QA personnel and customers. To reduce the coordination problems in this area, the company has been using a web-based commercial tool called VersionOne (http://versionone.com) which has been providing an excellent project management mechanism for both sites. Through VersionOne, developers at both sites can actually see how each project is divided, what projects are going on, the status of each project, who is working on each project, and when those projects are expected to be completed.

Recently, the company picked another commercial tool called JIRA (http://www.atlassian.com/software/jira/) to track information mainly on what kinds of bugs exist, who is working on each bug, and the status of each bug. The JIRA actually replaced an old tool called MOM, which was developed internally, and was used to create information for support problems. However, using JIRA has some downsides as stated,

But it (JIRA) also has drawbacks. We’ve now broken up for instances, the fixed description, the problem, various fields, and who’s worked on it. But in JIRA, you can’t search. If you are looking for some word maybe in the problem somewhere, you have to search each and every individual field. So, if you think, for instances, has some value in there other than the comments and summary and description, you must put the value in each field and then search individually. And searching is not near as good as MOM has, for instance, you can only search on whole words. You have to have the beginning of word. You can’t search for the value within a word.

Despite the downsides, JIRA has been providing developers with a useful bug tracking mechanism for both sites. JIRA also has a special code called severity or priority code which enables support people to automate the choice on what bugs to work on next. It seems VersionOne and JIRA play a vital role in the company to reduce coordination problems that can arise in tracking and managing projects and bugs.

**Communication Issues**

It is well known that most failure in software products results from ineffective communication (Parnas, 2006). It seems the company has realized and understood the important role of communications in the software development process because they have put a lot of effort toward establishing a good communication channel between the two sites. For example, the company has set up expensive devices to allow video conference between the two sites. However, due to the time and use constraint, the video conference was not always available for the individual software developers. Rather, the video conference was usually used for team meetings between sites once or twice a month. Because of the unavailability of video conferencing, phone conferencing was used more often. However, it seems that phone conferencing has problems as well. According to Jenny, a lead engineer,

When we have the daily Scrum meeting in the morning we did it in a couple of different ways. One was video conferencing and that was actually quite good. We also just did phone conferences which were not as good. Mostly because when somebody made a comment, their faces didn’t make the background. And of course the Scrum master couldn’t do anything about that. But that was counter-productive. If you have two groups going in video conferencing, it’s much better. You can see face-to-face and if somebody is upset, you know about it. You know making faces is not being able to hear over the phone. That did happen.

Other developers also mentioned in the survey that video conferencing helped them to see the other person’s facial expressions, gestures, and body language. But one developer stated that “it’s still hard to know whether they understand what I am explaining to them”. Though video conferencing is much better than phone conferencing, there are still limitations to using video conferencing. The daily Scrum meeting and the daily Scrum of Scrums meeting have been mostly done through phone conferencing and the monthly sprint planning and review meeting have been done through video conferencing. Other than video and phone conferencing, developers have been using a tool called web-demo (http://www.beamyourscreen.com) to show people in remote sites things on someone’s screen. This tool together with a phone system seems to work very well as a developer mentioned,
Actually we used web-demo which worked out very well for us when we are going to do code reviews and we used that web demo and phone conferencing. And it was like being in a media-room, actually it was much better than being in a media-room in a lot of cases because we were able to scroll to where we want to look at. It was very efficient.

The last multi-media that the company has been using are an instant message (IM), email system, and virtual private network (VPN). All of developers mentioned that they were using the instant message and email system much more than before they had been divided into two groups. It seems the instant message system is used more effectively than the email system for a short and quick question and answer. The instant message system also was more used by new employees. Several developers mentioned,

Yea, a lot more when you are working with a new person, you are using it (IM tool) all the time because he’s asking questions back and forth over an IM session. You are using it a lot with a new person where if he would have been here, he would have come in the office to show to him instead email him back and forth, or you would have been looking at it together. You could have used web demos to do that but in most cases it is short questions. We didn’t feel like we had to do something like that.

One developer who used to use VPN said in the survey question “Sometimes the VPN is too slow which causes frustration”. It was obvious that people did not use VPN once they found it was not fast enough. Though the video/phone system, web-demo system, and instant message/email system help software developers reduce the geographic distance between two sites, it seems those multi-media systems have not been working as optimally as face-to-face conversation. Several developers said “It would so much easier to be able to work with somebody who is right here in my office.” Mike also mentioned, “Well, in some ways it would be better to be down there at times.” He went on to explain that,

It hasn’t been that big of a disadvantage but once in a while, we just get two or three guys together, if there’s an issue and somebody wants to talk about options or kind of brainstorm on the idea. For me to join that group from this remote site is more difficult.

The company seemed to suffer a lot when a communication line was down because of heavy dependence on it. Mike mentioned, “We had a few problems last month with phone systems not working very well. When the phone systems were down it was a big deal plus when phone systems were down the data line went down too.” Because the telephone systems and data systems share the same physical line if the physical line was not operational, employees could not do anything.

The sixth principle of agile manifesto declares that “The most efficient and effective method of conveying information to and within a development team is face-to-face conversation” (http://agilemanifesto.org/principles.html). This principle tells us that team members should be collocated for an efficient and effective information sharing. As detailed in this section, however, under the distributed Scrum environment, this principle should be changed to find a different solution, and changes should be applied to the practices of Scrum including the daily Scrum meeting, the daily Scrum of Scrums meeting, the sprint planning meeting, and the sprint review meeting.

**Control Issues**

One of the interesting aspects of the Scrum method is the ability to see the visibility of the progress of each project every thirty days. Project managers in the company usually break up items that need to be done and it is a short enough time that if they need to insert something into development, it can easily be added without a problem. However, the project managers break down the tasks in what they think could be completed in thirty days and all the items on the backlog are consistently not completing. One project manager said,

Some things just keep getting carried over to the next month. There does not seem to be any responsibility on the developer side to complete the task at both sites. If they do not, it is o.k. and they hope to do better the next month in estimating their project. In the daily scrum, it seems to lack some of the “what did you do yesterday” accountability in it. And taking on specific tasks for that day to work on is usually too generic. I believe we have become too relaxed on what we accept and so that encourages some projects to drag over time. When we meet with the management team at the end of the month, no one seems to take the responsibility for those items not completed. The project managers do not have any control on getting people to work faster or harder, so it is a little frustrating.

It is a downside of the DSD that developers do not have the ownership of completion because of lack of relevant control in both sites. Batra et al. (2006) suggests setting up a coordinator in one site and ambassador in the other site to ensure the control. Similarly, under the distributed Scrum, if would be helpful if either project managers or Scrum masters could have control on getting developers to work faster and harder.
Training Issues

New employee training issues emerged when the information and knowledge-sharing issues and communication issues were brought up. Mike and Jenny mentioned that one of the biggest problems that the company has been facing is new employee training. Because of the complex nature of the software program that the company has been creating, new employees need to be trained for an extended period of time. It seems the problem gets worse when an employee who has expertise in one field is at the remote site and the new employee needs to learn the new field from a remote employee. In this case, it seems that the training through video/phone conferencing or web demo has limitations. A developer stated “You have to have face-to-face and one-on-one training in order to make it efficient”.

Trust and Confidence issues

Trust and confidence issues arose when a Scrum team member in one site asked a Scrum team member in other site to do a certain task but no progresses on the task are visible. Issues also arose between developers in one site and a Scrum master in other site who is supposed to do administration work and help developers focus on their work by providing what they need. But a Scrum master who did not remove the developers’ impediments was a problem as a developer mentioned,

We need somebody who is going to be able to get you what you need and somebody telling me that “You’re going to get your information back.” And we weren’t getting any information back and things weren’t followed up.

It seems developers in one site had a hard time to establish a feeling of trust with a Scrum master or developers in the other site if they did not complete a certain task and the lack of trust led to a reduction of confidence as a developer stated,

People in one site didn’t trust people in the other site if they were not able to get the job done. I think that was one of major problems we had. That really hurts our confidence. I need to be able to have confidence level. I think it’s really important to know who you are talking to and what level of information they have.

Another developer also mentioned that “I work better with people who I know better so I understand what they are really saying.” The fifth principle of agile manifesto (http://agilemanifesto.org/principles.html) also says “Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done”. This implies developers should work in environment that they want and should have support that they need, and at the same time they should give other team members trust to gain a high confidence level. It would also be helpful if each member of the Scrum team in both sites could take the time to get to know other members.

DISCUSSION

The key issues found in this study suggest that the role of a Scrum master should be adjusted. Since the Scrum master is the key person for the success of Scrum (Schwaber and Beedle, 2002), the Scrum master should be engaged more actively in projects as a coordinator and also a controller in distributed Scrum environment. For example, as a coordinator/controller, the Scrum master needs to ensure that the information and knowledge are shared well between the sites and that the tasks are divided and assigned well through the sprint planning meeting. Also, the Scrum master needs to have the authority to motivate developers to work hard and to take ownership of the projects. These tasks can be eased by using various tools such as Wiki, VersionOne, and JIRA.

The various Scrum meetings should also be adjusted due to the geographical distance. For example, in daily Scrum meetings which are held every day for less than 15 minutes, team members come to talk about what tasks have been done since the last meeting, what tasks will be done before the next meeting, and what the issues and challenges are imposed on the tasks. Scrum team members come to daily Scrum meetings to communicate with other team members to find out what is going on. However, the effectiveness of communication between the sites is severely limited compare to face-to-face conversation. To mitigate the problem, the daily Scrum meeting and other Scrum meetings should be held with good communication devices. Among many different multi-media devices, a video conferencing system between the sites is recommended as the best way to communicate. Other tools including remote desktop, an email system, an instant message system, and a phone system can mitigate the communication problems too.

CONCLUSION

This paper presents how key components of the Scrum method should be adjusted in distributed environment and what issues arose if large-scale and mission-critical projects are carried out using the distributed Scrum. Also, the paper explains the potential solutions for each issue and introduces the technologies used in company in the case study to mitigate the problems.
Though the company has put a lot of effort towards removing or minimizing the issues that resulted from the geographic distance of the two sites, there are still some issues that the company must resolve. As mentioned before, establishing well-structured information and knowledge-sharing systems is a rudimentary process for the company. Establishing good communication channels using various forms of multi-media should also be proceeded before successful software development. Having good coordination between developers and QA personnel, and bestowing authority to Scrum masters to ensure control in both sites is highly recommended. It appears that the company must find more efficient and effective way to train new employees so that they can be assigned to write code in a complex field with full knowledge of that area. Also, the company needs to encourage Scrum team members in both sites to get to know each other better. If the Scrum team members between two sites can develop a feeling of trust and raise the confidence level, they can work together better. If all of suggestions are fulfilled the distributed Scrum can be viable for large-scale and mission-critical projects.
REFERENCES


Cao, L., Mohan, K., Xu, P. and Ramesh B. “How extreme does extreme programming have to be? Adapting XP practices to large-scale projects,” Proceedings of the 37th Hawaii International Conference on System Sciences (HICSS’04), Hawaii, May 2004.


