An Empirical Study on the Impact of Scrumban on Geographically Distributed Software Development

Ahmad Banijamali, Research Dawadi, Muhammad Ovais Ahmad, Jouni Similä, Markku Oivo and Kari Liukkunen
Department of Information Processing Science, University of Oulu, Oulu, Finland

Keywords: Scrumban, Agile, Lean, Kanban, Distributed Software Projects, Coordination, Software Factory.

Abstract: Coordination in software projects is a challenge, particularly when it comes to distributed software development (DSD). Agile software development is a well-known paradigm for improving software development; however, there is little understanding of its impact on DSD projects. This paper describes an empirical study conducted within two Software Factory settings in Finland and Italy to investigate how Scrumban can impact coordination in geographically distributed software development. It provides the results from a project case that applied Scrumban to develop a web-based application for Time-banking. This study shows that aspects of Scrumban, such as iterative development, enhanced visibility, and limited work-in-progress, would alleviate the challenges of leveraging resources, synchronization between distributed sites, communication, and culture. It also explains that technical and security issues in the coordination of distributed projects may demand for solutions other than Scrumban.

1 INTRODUCTION

Schwaber and Sutherland (2012) argue that Agile projects are successful in the rate of 42% of cases, which is considerably more than what has been achieved with waterfall model (14%). Agile methods are iterative and incremental, in that collaboration between self-organizing cross-functional teams provides requirements and solutions (Alam and Chandra, 2014). We have selected Scrum for this study, as it is the most frequently used Agile method in software development (Rodriguez et al., 2012).

On the other hand, Kanban has not been widely adopted in software development (Mahnic, 2014). In 2004, Kanban entered into the Agile realm when David Anderson introduced it in practice while assisting a software development team at Microsoft (Ahmad et al., 2013). The high expectations for Kanban are the result of its adaptability toward changes in requirements, the visualization of project processes, and its role in increasing communication and cooperation among team members (Kniberg and skarin, 2010).

There seems to be a concern regarding the combination of Kanban and Agile practices. Scrumban (Scrum and Kanban) applies Scrum as a prescriptive method, while it encourages process improvements through Kanban to allow projects to continuously improve their processes (Khan, 2014). According to Ladas (2009), Scrumban is appropriate for teams that are already using Scrum.

Scrum does not consider the organization as a whole during its implementation (Rodriguez et al., 2014) and has limitations, such as lack of work visibility and changing task priorities (Tripathi et al., 2015). These limitations can be mitigated by using Kanban alongside Scrum. Scrumban thus inhibits the characteristic to embrace change and help to establish better relationships between business and information technology departments (Auerbach and McCarthy, 2014).

Geographically distributed teams with poorly planned coordination often end up with unmatched deadlines, costs overrun, and even cancelled projects (Smith et al., 2005). There are additional challenges that can lead to complexities with respect to location, time, culture, and language in distributed software development (DSD) (Gupta and Fernandez, 2011). The idea of utilizing manpower from different locations is tempting, but it creates excessive coordination tasks in projects. It needs to be ensured that everyone has clear idea of the project goals and

Banijamali, A., Dawadi, R., Ahmad, M., Similä, J., Oivo, M. and Liukkunen, K.
An Empirical Study on the Impact of Scrumban on Geographically Distributed Software Development.
DOI: 10.5220/0005686405670577
In *Proceedings of the 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD 2016)*, pages 567-577
ISBN: 978-989-758-168-7

is committed to achieve them.

Šmite et al. (2010) discuss the concept of applying Agile methodologies in the context of DSD. They compare the characteristics of Agile and DSD and argue that communication in Agile projects is informal, face-to-face, and synchronous, while DSD projects require formal, computer-mediated, and often asynchronous communication. Moreover, Agile projects apply change-driven and self-managed coordination and light-weight control; however, DSD settings need plan-driven and standardized coordination among sites, which is achieved through several command and controls. Despite their opposite characteristics, the combination of Agile and distributed development is of high interest to companies (Šmite et al., 2010).

Geographically distributed development, in itself, is a vague term because there can be different types of distributed teams based on the time difference between the involved teams (Carmel and Espinosa, 2011). Two configurations of distributed teams that can be taken into consideration are North-South and East-West. North-South distributed teams are a combination in which teams do not have a considerable difference in time zones, while the East-West configuration involves a significant time zone difference (Carmel and Espinosa, 2011). Our investigated software factories (described in section 3.1) were distributed from the north to south of the Europe; hence the East-West setting is beyond the scope of this research.

According to Šmite et al. (2010), there is limited research and understanding about the application of Agile methodologies in DSD. In addition, Scrumban is a new development approach in the software engineering domain, and existing literature provides little information on Scrumban's impact on DSD projects. Increasing interest in globally distributed software development practices has motivated us to investigate the following question: "What is the impact of Scrumban methodology on geographically distributed software development projects?" As coordination among developers is the critical issue within those environments, we have mainly discussed Scrumban from this perspective.

The remainder of this paper is structured as follows: Chapter 2 presents an overview of previous research on Scrumban and its impact on software project environments. Further, it elaborates geographically distributed software development. Chapter 3 introduces the Software Factory settings and project case used for this research, the project coordination model, and the applied research approach. Chapter 4 presents findings of our study, the limitations, and direction for future studies. Chapter 5 concludes the paper and highlights the main contribution of our work.

2 RELATED WORKS

This section summarizes the literature with respect to the Scrumban and DSD practices.

2.1 Scrumban

Scrum is an incremental Agile software development methodology. It operates through a series of iterations that require continuous planning, defined roles, and project artefacts (Schwaber and Beedle, 2002; Schwaber, 2004). Scrum is the most frequently applied Agile software development method (Rodriguez et al., 2012) to achieve small but continuous deliverables. It facilitates regular feedback after each iterative development process, called a "sprint" (Nikitina and Kajko-Mattsson, 2014). Rising and Janoff (2000) have pointed out that Scrum is beneficial, particularly for projects in which all the requirements are not clear in advance and some type of chaos is expected during the project.

Kanban is a relatively new concept in the field of software engineering that was originally applied in Lean manufacturing (Ahmad et al., 2013). While Scrum focuses on one iteration (called a sprint) at a time, Kanban supports a continuous workflow (Mahnic, 2014). Kanban provides the flexibility of managing the workflow within teams. It limits the work in progress (WIP) in each activity to a maximum number of tasks or items at any given time. Moreover, it does not suggest strictly defined roles and sprints (Nikitina et al., 2012). It provides a clear visualization of the phases in the project lifecycle.

By combining Lean and Agile methodologies, project members can receive fast and iterative feedback while they have the ability to implement the necessary changes and respond to the feedback. The combination of Agile and Lean in co-located projects enhances coordination among team members, increases team morale, and produces better outcomes (Auerbach and McCarthy, 2014). Lean increases the scale of the development process and makes it efficient, while Agile principles help to make the process flexible (Rodriguez et al., 2014).

Both Scrum and Kanban are similar in the sense that both improve transparency, aim to release software as soon as possible, work on the principle of breaking work into pieces, and continuously optimize the project plan (Barash, 2013). Ladas (2008) has argued that if Kanban is used alongside Scrum, they both can complement each other. Scrumban incorporates the iterative planning of Scrum but is more responsive and adaptive to changes in user requirements. Project members who have had good experience with Scrum can benefit from Scrumban, as it improves their knowledge and capabilities (Ladas, 2008). By combining Scrum and Kanban, researchers hope to create more flexibility in projects as well as iterative paces that Scrum has provided (Ladas, 2009).

Table 1 reflects the key points of using Scrum and Kanban in the same project by showing several examples. In section 4.1, we will use these points for our analysis in the context of distributed software development.

Table 1: Scrum and Kanban methodological elements.

Ref.	Study Place	Key Points
Nikitina et al. (2012)	Vietnamese office of a Swedish software development company	Scrum: Iterative and incremental Regular feedback Strict roles and rules Kanban: Visualization Limiting WIP Scrumban: Self-organizing Collaborative teamwork
Mahnic (2014)	Faculty of Computer and Information Science, University of Ljubljana	Scrum: Incremental and iterative Planned project Regular feedback Kanban: Maximize workflow Visualization Limiting WIP
Joshi and Maher	Arrk Group, a multinational software development company	Scrumban: Limiting WIP Optimal resource utilization Collaborative teamwork Quick decisions Customer satisfaction
Brinker (2014)	GoGo, a company that offers services such as internet, entertainment, text messaging, voice, etc. in the aviation market	Scrumban: Visualization of workflows Transparency Increased team participation

One factor that Scrumban inherits from Kanban is the visualization of workflows (Khan, 2014). Scrum completes tasks through sprints that are already planned, but Scrumban allows more flexibility and planning only for following sprint. This helps projects to limit the WIP. When the limit of tasks in a particular workflow is reached, team members help each other to complete the tasks in that workflow rather than starting a new one. This increases the coordination among team members and also reduces the possibility of bottleneck (Khan, 2014).

Scrumban, unlike Scrum, has no strict rules and roles and encourages self-organized teams. As a result, team members manage their tasks by themselves and make quicker decisions. Khan (2014) argues that Scrumban reduces the relevant tasks of planning for the whole iteration (the same as Scrum), as meetings are set only when required and tasks are changed depending on the output of the ongoing sprint.

The implementation of Scrumban presents several challenges as well. The flexibility regarding production changes can cause new challenges in, for example, assigning resources and project time-tables. Since Lean methodology calls for considering the whole organization through implementation (Karvonen et al., 2012), the combination of Kanban and Scrumban increases the complexities of planning for the whole organization activities. Moreover, it is not always possible to include business personnel or management executives to develop project backlogs or receive regular feedback (Rodriguez et al., 2014).

2.2 DSD

DSD that addresses global practices for producing software is applied through multi-geo, multicultural, and multi-temporal environments. Distributed development practices benefit from lower costs, enhanced performance, and less time to markets (Sutanto et al., 2011).

Prior studies (Nakamura et al., 1997; Jiménez et al., 2009; Šmite et al., 2010) have addressed the significant challenges of distributed environments in terms of communication gaps between multiple sites, group awareness, software configuration management, knowledge management, flexible coordination, collaboration, project management, process support, tools support, quality management, and risk management.

Coordination is a pressing issue in global software development. People at the research and development center of Yahoo in Norway mentioned that the time zone difference was a major cause of problems when dealing with dislocated teams (Carmel and Espinosa, 2011). Noll et al. (2010) argued that the main barriers to coordination in distributed projects are geographic, temporal, cultural, and linguistic differences. They proposed that project teams should enhance site visits, use synchronous communication technology, and apply knowledge-sharing infrastructure to transform the implicit knowledge to explicit (Noll et al., 2010). Other scholars (Mak and Kruchten, 2006; Redmiles et al., 2007; Sidhu and Volberta, 2011) have argued that coordination issues come from (1) a lack of flexibility and integration, (2) poor role support, (3) decreasing informal communication and workplace transparency, and (4) limitations imposed on formal communication. Therefore, it is necessary to apply the most suitable methodologies and tools to improve the coordination of interdependent tasks in distributed sites.

There are several instances of the application of Scrum in distributed development projects (Sutherland et al., 2009; Šmite et al., 2010; Carmel and Espinosa, 2011; Schwaber and Sutherland, 2012).

The American software consulting company Agile Factori implemented a successful software development project using Agile methodologies. The project was provided by “Big Oil,” an American company consisting of 4 teams in which two were located in America and the other two in Brazil and Argentina. All four teams had a real-time video screen with audio that showed activities at the other sites. In addition, one screen at each site showed a dashboard of in-process software components. This allowed other sites visualization, increased awareness, and better coordination among teams (Carmel and Espinosa, 2011).

SirsiDynix (U.S) (Sutherland et al., 2007; Sutherland et al., 2009) has successfully implemented distributed Scrum since 2005. Using distributed Scrum, SirsiDynix collaborated with the Russian company Exigen in 2005 for a large project (Sutherland et al., 2009) employing more than 50 members in total and producing over one million lines of code. The output of this distributed team was estimated to be equivalent to the work of a 350 co-located-person team working in a waterfall model (Sutherland et al., 2009).

An international Agile software development company, Xebia, located in France, India, and the Netherlands, had also implemented Scrum successfully during 2006–2008 (Sutherland et al., 2009). Distributed Scrum was used alongside XP programming in multiple projects by Xebia, and the results showed that the distributed teams were as effective as co-located teams. These instances show that globally distributed teams can be as productive as co-located teams effectively applying Scrum (Sutherland et al., 2007; Paasivaara, 2011).

3 RESEARCH PROCESS

3.1 Project Case and Software Factory Settings

Software Factory settings provide developers with a development setting consisting of domain-specific tools that help to transform abstract models into implementations (France and Rumpe, 2007; Abrahamsson et al., 2010; Ahmad et al., 2014). Through Software Factory settings, reusable development practices such as patterns, models, guidelines, and transformations are accessible from the viewpoint of a specific aspect in the development context. This enables domain-specific validation and guidance delivery (Greenfield et al., 2004).

A joint five-month software development project called T-Bix was initiated between the University of Oulu, Finland and the University of Bolzano, Italy in their respective Software Factories. The aim of the project was to develop a web-based application for time-banking to be operational in South Tyrol in Italy. The web application was required to possess the facilities of searching, posting, and applying for jobs and skills for unemployed and elderly people who were interested in being part of the time-banking community. Since T-Bix project teams were located in Europe (North-South DSD configuration), they did not experience drastic temporal differences; however, the long physical distance and diverse cultures, languages, and social behaviors remained challenges in the project.

The Finnish team members were comprised of one PhD candidate and four master’s degree students who were working locally in Oulu. The team from Italy had a Software Factory coordinator with a PhD degree and four master’s degree students. A member of the Italian team was working remotely from Lithuania. There was one student on each team with industrial experience; however, the rest of the teams did not have prior experience in industry. Each team was comprised of one project manager and three developers. There was an Italian business customer who was in direct contact with both teams. The customer communicated his needs through meetings and emails; teams attempted to interpret the customer’s requirements into the user stories and backlogs. Scrumban was the methodology implemented in Finland, and Scrum was used in Italy.

Teams communicated frequently via collaboration tools like Google Hangout and Skype to discuss and verify project requirements, progress, deliverables, challenges, and deadlines. After each sprint, teams presented the respective deliverables and progress and received feedback from both the customer and other team members. They also planned for the next sprint.

Carmel and Espinosa (2011) noted that identifying the best time for meetings is a major concern in distributed projects. T-Bix project meetings were scheduled with respect to the temporal difference between Italy, Finland, and Lithuania (there was a developer from the Italian team who was working from Vilnius). To have other site and customer involved, the meetings were often held in the afternoon (Finnish time zone). This is an advantage provided by North-South collaboration in that meetings could be held in the daytime and not much time shifting is required.

The frontend of the application was developed with direct contact with the customer in Italy. The backend, including the database development and integration of the frontend and backend, was developed in Finland. The codes were shared on GitHub (https://github.com), where some feedback and comments were also shared.

An identical Kanban board was created in JIRA (https://www.atlassian.com/software/JIRA) by the Oulu team and shared with the team members in Bolzano. The Kanban board was updated regularly, providing visibility of the board and tasks across both teams.

In addition to JIRA boards, the Software Factory in Oulu was equipped with physical Kanban boards utilized throughout the project's lifetime. The boards were divided into four sections: backlog (features), to do, in progress (WIP), and done and consisted of user stories planned in each sprint. Once each sprint was completed, the Finnish team updated the boards with new tasks and shifting completed jobs to the "done" section. Figure 1 shows a snapshot of the board.

3.2 Project Coordination Model

The project was proposed by the customer to the University of Bolzano with the aim of decreasing the rate of unemployment in South Tyrol. Subsequently, the University of Bolzano had the idea of making the project a distributed Software Factory project between the two universities.

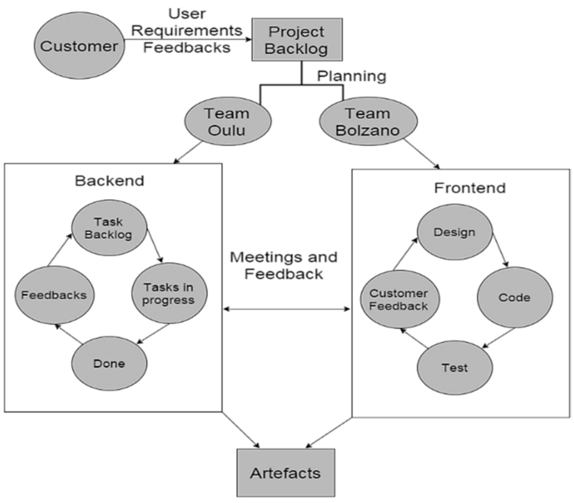

Figure 1: A physical Kanban Board in Oulu Software Factory.

The customer was in contact with the teams with respect to the elicitation of requirements, acceptance testing, and the validation of artefacts. The user interface of the website was designed and validated through regular meetings with the customer. The codes and designs were continuously uploaded in GitHub, in which both teams updated their last works. The next sprint was planned according to the feedback and suggestions made by the customer and both teams. The following model (Figure 2) shows how project was carried out among the teams.

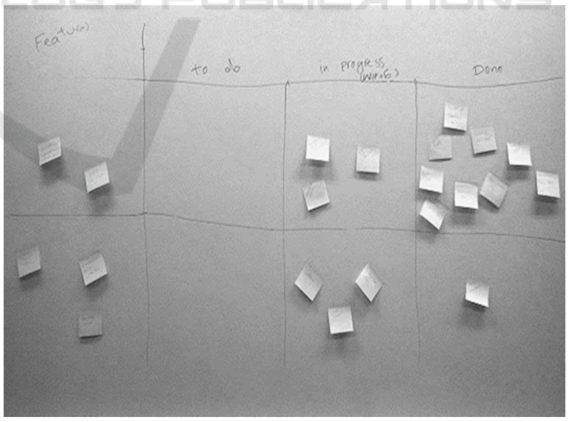

Figure 2: Project coordination model.

3.3 Research Approach

This study exploits empirical software engineering methods. The authors have applied semi-structured interviews to collect the empirical data from the project members. The participants of this empirical study are members of the Oulu Software Factory who were interviewed after the project's completion. Four rounds of interviews were conducted, which lasted from 45 minutes to 2 hours. All interviews were recorded and transcribed, in which authors could analyze them based on the needs of this study.

A semi-structured interview format was preferred, as it provides a clear set of instructions for the interviewer, who usually follows a paper based interview guide during the interview. The availability of questions beforehand makes the interviews easier for the interviewer and the openness of this type of interview provides the interviewees with the freedom to express their views using their own terms. In addition, the comparable qualitative data obtained from semi-structured interviews is regarded as reliable for analysis (Cohen and Crabtree, 2006). Table 2 summarizes the roles, empirical experiences, and expertise of the interviewees.

Table 2: Interviewees' backgrounds.

Intervi-ewee	Role in the project	Empirical exp.	Expertise
D1	Project manager	>10 years	Proj. mgmt. UI Design JIRA, GitHub
D2	Programmer	---	PostgreSQL, JIRA, GitHub
D3	Programmer	---	PostgreSQL, Java, JIRA, GitHub
D4	UI Designer	---	UI Design, Java, JIRA, GitHub

4 RESULTS

This section summarizes our findings regarding Scrumban's impact on the T-Bix project as well as the limitations and opportunities for future research.

4.1 Findings

Table 3 explains how the impact of Scrumban has been realized in the coordination between North-South distributed sites.

For this purpose, we have investigated the top issues in DSD projects that have been already introduced by other scholars (Nidiffer and Dolan, 2005; Espinosa et al., 2007; Barcus and Montibeller, 2008; Carmel and Espinosa, 2011). This section reviews the impact of key aspects of Scrumban (provided in section 2.1) on DSD issues.

Strategic issues within DSD settings are concerned with the difficulty in leveraging available resources. Issues should be identified carefully in which stakeholders can anticipate and manage risks (Nidiffer and Dolan, 2005). Since T-Bix was an evolutionary project done through iterative sprints, teams were able to find new ways to leverage available resources and skills. Within the initial meetings, two teams discussed the experience and expertise of their members, clarifying how the project resources were divided between the two sites and how the project duties should be assigned.

However, the team members mentioned their increasing responsibility during the later sprints of the project. The project manager [D1] explained that they were asked to accomplish some additional work on coding. Adapting to these workflow changes made it difficult to complete the project. A developer [D4] explained that after much discussion, the two teams decided to assign additional tasks to the Oulu team, as they had more technical skills:

"After much discussion, we had to accept more work, as Bolzano was not able to complete it. We should provide more deliverables at the end of the project. We had no choice because we wanted the project done."

The teams applied JIRA to establish the project's structure and define the roles of the two sites. Project tasks were assigned to the teams members according to their roles and skills. Furthermore, JIRA created visibility in the WIP for each role compared to other developers. The project manager [D1] confirmed this:

"Using JIRA, I could monitor the progress of different completed tasks with respect to the roles. It provided me an opportunity to recognize the tasks that required extra coordination."

Project and process management in DSD involves discussing problematic situations in synchronizing work between distributed sites (Barcus and Montibeller, 2008). Integrated quality, shared workspaces for storing files, and engineering tools are potential enablers of this issue. The complexity also arises from the fact that there should be sufficient communication between two teams before they can prioritize project tasks and decide which one is to be carried out by which team, as in the case of the T-Bix project discussed. The teams had agreed upon a preliminary division of work, but additional tasks were later added to the project by the customer. The members mentioned that the added tasks caused several challenges in managing their ongoing tasks. To control the scope of project, the involved teams should manage changes in a planned way. Any changes in the project scope may affect the priority and division of work among the sites. The project manager [D1] declared the following primary decision criterion for allocating tasks between sites:

"Consistency between the requested feature and available skills and knowledge at the sites was our decision criterion for allocating tasks to sites."

Using Kanban boards in JIRA improved the visualization and transparency on the completed, ongoing, and planned tasks. One developer [D3] mentioned the following:

"The Kanban board in JIRA was quite helpful because we could not frequently update the pictures of the physical Kanban board for the other team. We applied JIRA's Kanban board to share the tasks we had completed and planned to do."

Using JIRA and GitHub, project members received feedback on their jobs, for example, for the codes that were uploaded in GitHub. One of the developers [D2] stated the following:

"For example, when the scripts in the database had problems, one of the programmers in Bolzano was using GitHub to send feedback regarding the issues and asking for solutions."

Another developer [D4] also believed the following:

"JIRA is a tool developed for task management purposes, but you cannot upload all project deliverables into it. It is not a shared platform, so we needed to use other tools, in which we could share other data."

Communication issues are related to the lack of effective communication mechanisms. It is very important to convey information such as the current state of the project as well as project challenges, schedule, and cost. In the case of distributed projects, communication plays an important role in collaboratively planning the project stages. Along with formal communication, informal communication between team members and with stakeholders can ease the working environment and develop coordination among them (Barash, 2013).
Applying Scrumban in DSD projects demands for both formal and informal styles of communication.

Table 3: Impact of Scrumban on coordination in DSD environments.

Scrumban aspects	Issues in distributed software development					
	Strategic	Project and process management	Communication	Cultural	Technical	Security
Iterative and incremental development	Highly improved toward latest sprints	Highly improved toward latest sprints	More sprints, more smooth	More iterations, fewer challenges	Slightly improved	No evidence
Predictable and well-planned project	No meaningful impact on leveraging resources at the other site	More iterations, more improvement	Effective communication for the planned tasks	No evidence	Slightly improved toward latest sprints	No evidence
Transparency	Positively impacted leveraging resources at both sites	Positively impacted task management within sites	No evidence	Slightly reduced challenges	No evidence	No evidence
Regular feedback	Slightly improved	Positively impacted task management within sites	Demands of both formal and informal feedback	Improved toward latest sprints	No evidence	No evidence
Limiting WIP	Positively impacted resource management	Decreased relevant challenges slightly	No evidence	No evidence	No evidence	No evidence
Self-organizing	Slightly improved	Positively impacted task management within sites	Improved informal communication	No evidence	No evidence	No evidence

Informal communication facilitates project implementation; however, the other type of communication creates a disciplined environment, which is necessary for coordination in DSD sites.

Communication was regarded as an important tool to ensure that the T-Bix teams were placed at the same level of understanding regarding the project. A developer [D4] highlighted the following:

"The Bolzano team had their own understanding of the project and we had ours. We had discussions to resolve the discrepancies and create balance between the two teams. Scrumban provoked us to have regular meetings with team members as well as the customer. This increased the level of communication in the project." However, the project manager [D1] mentioned that different time zones created little discomfort for arranging meetings.

Scrumban leads to a great deal of communication. One developer [D4] argued the following:

"At first, we had many problems in our communication because the project members complained that the other site hindered the project's progress and was not completing its tasks well."

However, finding new communication channels as well as more effective planning in the project led to a higher level of communication between teams. It was claimed that:

"We had many challenges in our discussions, but since people have had more communication and became increasingly more acquainted with the way the other team works, communication became smoother."[D1]

Cultural issues involve the conflicting behavioral processes and technologies (Nidiffer and Dolan, 2005). Different socio-cultural backgrounds make communication more complicated regarding the lack of understanding about other social behaviors, cultures, and languages. The T-Bix project shows that people have different expectations regarding working in multinational teams; for example, one developer [D2] explained the following:

"It was quite good for distributed software development to include multiple cultures. It was interesting to work with people with different backgrounds." However, other people found multi-cultural settings more difficult than co-located projects.

Due to the nature of Software Factory projects, team members were completely new to each other and they were assigned to this project with no prior knowledge of the other team members. At the beginning of the project, they had several challenges in communicating with each other and establishing a good organization for their project; however, the evolutionary development as well as receiving feedback on the requirements and skills alleviated cultural barriers when people met for several sprints.

Technical issues in DSD environments are related to incompatible data formats and exchanges. Creating standards and web services could be seen as potential enablers to resolve this issue. T-Bix shows that during different sprints, teams progressively realized the technical facilities and needs of other sites. The iterative nature of Scrumban helped them to meet those needs and prepare to meet the internal project standards and agreements.

Security, on the other hand, involves ensuring electronic transmissions' confidentiality and privacy (Nidiffer and Dolan, 2005). It can be improved through emerging standards for secure messaging. The T-Bix project did not provide meaningful evidence of Scrumban's impact on improving security issues in DSD settings. However, this study was conducted with respect to coordination issues and other project issues are beyond its scope.

4.2 Limitations

Scientific studies on Scrumban are very limited. In this research, we investigated the impact of Scrumban on coordination in geographically distributed development. We believe the results can be generalized to other distributed projects using Scrumban, as even though our research was conducted in Software Factory settings, the results are based on a real business case with a real customer outside the university environment.

The coordination among teams might have been different if the distributed teams had an East-West configuration. Finding a suitable time for meetings, on-time responses to emails, and other queries would have taken more time. With a greater time zone difference, it is fair to say that teams would have possessed greater variance in their work cultures; for example, from India to the U.S. However, East-West teams could likely work with ease and spend more time on their decisions and responses to emails and queries.

The Software Factory project was small with a limited number of interviewees. However, several studies have reported the benefits of using students as the empirical research subjects (Höst et al., 2000; Madeyski, 2009); we assume the level of impact of Scrumban may differ in a larger DSD project. In addition, industrial projects may provide more evidence on the effect of Scrumban on DSD projects.

To avoid inappropriate interpretations, we have presented the interviews the same as the project members described. However, the authors' judgment could not be completely eliminated.

4.3 Future Research

The results of this study could be utilized to predict Scrumban's impact on coordination in distributed projects. However, it would be interesting to see how Scrumban supports large and industrial distributed projects.

In this study, we have discussed coordination in DSD; thus, other issues in these kinds of settings still remain untouched. Future researchers can investigate the impact of Scrumban on other aspects of globally DSD projects. In addition, the impact of age, education, and years of experience etc. on the use of Scrumban in DSD projects can be another research topic.

Software Factory settings is an interesting concept to test new ideas and methodologies related to software development. To review the impact of Scrumban on East-West distributed teams, other Software Factories from different time zones should be included in future studies.

5 CONCLUSIONS

Combining the key concepts of successful software development methodologies (Kanban and Scrum), Scrumban is iterative as well as responsive to the changes in requirements of ongoing projects. Along with that, the fast and efficient approach of Scrumban makes it a favorable choice in the software engineering domain. The impact of Scrumban on software development projects, either co-located or globally distributed, has not been researched a great deal. We have investigated the impact of Scrumban on distributed sites within a Software Factory project.

Distributed sites have to receive changes in projects at the right time. Achieving this purpose, collaboration tools are necessary to share updated information and resolve the challenges. In addition, creating a defined organizational structure with specific roles creates visibility in project, which is necessary for coordination and task management in projects.

This study could effectively be applied in both academic and business environments. Academic studies could investigate other aspects (rather than coordination) of using Scrumban in DSD projects. Moreover, industrial projects can be efficient if the members are well informed about the challenges and strengths of Scrumban in different project settings. This will help them with the efficient planning of the project deliverables and interactions among the teams involved.

ACKNOWLEDGEMENTS

This research was supported by the DIGILE Need for Speed program, and partially funded by Tekes (the Finnish Funding Agency for Technology and Innovation). We would like to thank DIGILE and Tekes for their support and the University of Bolzano for its excellent collaboration.

REFERENCES

Abrahamsson, P., Kettunen, P., and Fagerholm, F. 2010. The set-up of a software engineering research infrastructure of the 2010s. In *Proceedings of the 11th International Conference on Product Focused Software,* pages 112–114. ACM.

Ahmad, M. O., Liukkunen, K., and Markkula, J. 2014. Student perceptions and attitudes towards the software factory as a learning environment. In *Global Engineering Education Conference (EDUCON), 2014 IEEE*, pages 422–428. IEEE.

Ahmad, M. O., Markkula, J., and Oivo, M. 2013. Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 9–16. IEEE.

Alam, S. S. and Chandra, S. 2014. Agile software development: Novel approaches for software engineering.

Auerbach, B. and McCarthy, R. 2014. Does agile+ lean= effective: An investigative study. *Journal of Computer Science and Information Technology*, 2(2):73–86.

Barash, I. 2013. Use of agile with XP and Kanban methodologies in the same project.

Barcus, A. and Montibeller, G. 2008. Supporting the allocation of software development work in distributed teams with multi-criteria decision analysis. *Omega*, 36(3):464–475.

Brinker, S. 2014. Using Scrumban (Scrum Kanban) for Agile Marketing - Chief Marketing Technologist, http://chiefmartec.com/2014/12/using-scrumbanlean-agile-marketing/.

Carmel, E. and Espinosa, J. A. 2011. *I'm working while they're sleeping: Time zone separation challenges and solutions*. Nedder Stream Press.

Cohen, D. and Crabtree, B. 2006. Qualitative research guidelines project.

Espinosa, J. A., Slaughter, S. A., Kraut, R. E., and Herbsleb, J. D. 2007. Team knowledge and coordination in geographically distributed software development.

Journal of Management Information Systems, 24(1):135–169.

France, R. and Rumpe, B. 2007. Model-driven development of complex software: A research roadmap. In *2007 Future of Software Engineering*, pages 37–54. IEEE Computer Society.

Greenfield, J., Short, K., Cook, S., Kent, S., and Crupi, J. 2004. *Software factories: assembling applications with patterns, models, frameworks, and tools.* Wiley Pub.

Gupta, M. and Fernandez, J. 2011. How globally distributed software teams can improve their collaboration effectiveness? In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on*, pages 185–189. IEEE.

Höst, M., Regnell, B., and Wohlin, C. 2000. Using students as subjects a comparative study of students and professionals in lead-time impact assessment. *Empirical Software Engineering*, 5(3):201–214.

Jiménez, M., Piattini, M., and Vizcaino, A. 2009. Challenges and improvements in distributed software development: A systematic review. *Advances in Software Engineering*, 2009:3.

Joshi, A. and Maher, S. Our Journey into Scrumban, http://www.arrkgroup.com/thoughtleadership/our-journey-into-scrumban/.

Karvonen, T., Rodriguez, P., Kuvaja, P., Mikkonen, K., and Oivo, M. 2012. Adapting the lean enterprise self-assessment tool for the software development domain. In *Software Engineering and Advanced Applications (SEAA), 2012 38th EUROMICRO Conference on*, pages 266–273. IEEE.

Khan, Z. 2014. Scrumban-adaptive agile development process: Using scrumban to improve software development process.

Kniberg, H. and Skarin, M. 2010. *Kanban and Scrummaking the most of both.* Lulu. com.

Ladas, C. 2008. Scrumban. *Lean Software Engineering-Essays on the Continuous Delivery of High Quality Information Systems.*

Ladas, C. 2009. *Scrumban-essays on kanban systems for lean software development.* Lulu. com.

Madeyski, L. 2009. *Test-driven development: An empirical evaluation of agile practice.* Springer Science & Business Media.

Mahnic, V. 2014. Improving software development through combination of scrum and kanban. *Recent Advances in Computer Engineering, Communications and Information Technology, Espanha.*

Mak, D. K. and Kruchten, P. B. 2006. Task coordination in an agile distributed software development environment. In *Electrical and Computer Engineering, 2006. CCECE'06. Canadian Conference on*, pages 606–611. IEEE.

Nakamura, K., Fujii, Y., Kiyokane, Y., Nakamura, M., Hinenoya, K., Peck, Y. H., and Choon-Lian, S. 1997. Distributed and concurrent development environment via sharing design information. In *Computer Software and Applications Conference, 1997. COMPSAC'97. Proceedings, The Twenty-First Annual International,* pages 274–279. IEEE.

Nidiffer, K. E. and Dolan, D. 2005. Evolving distributed project management. *Software, IEEE,* 22(5):63–72.

Nikitina, N. and Kajko-Mattsson, M. 2014. Guiding the adoption of software development methods. In *Proceedings of the 2014 International Conference on Software and System Process,* pages 109–118. ACM.

Nikitina, N., Kajko-Mattsson, M., and Strale, M. 2012. From scrum to scrumban: A case study of a process transition. In *Proceedings of the International Conference on Software and System Process,* pages 140–149. IEEE Press.

Noll, J., Beecham, S., and Richardson, I. 2010. Global software development and collaboration: barriers and solutions. *ACM Inroads,* 1(3):66–78.

Paasivaara, M. 2011. Coaching global software development projects. In *Global Software Engineering (ICGSE), 2011 6th IEEE International Conference on,* pages 84–93. IEEE.

Redmiles, D., Van Der Hoek, A., Al-Ani, B., Hildenbrand, T., Quirk, S., Sarma, A., Filho, R., de Souza, C., and Trainer, E. 2007. Continuous coordination-a new paradigm to support globally distributed software development projects. *Wirtschafts Informatik,* 49(1):28.

Rising, L. and Janoff, N. S. 2000. The scrum software development process for small teams. *IEEE software,* (4):26–32.

Rodriguez, P., Markkula, J., Oivo, M., and Turula, K. 2012. Survey on agile and lean usage in Finnish software industry. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement,* pages 139–148. ACM.

Rodriguez, P., Partanen, J., Kuvaja, P., and Oivo, M. 2014. Combining lean thinking and agile methods for software development: a case study of a Finnish provider of wireless embedded systems detailed. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on,* pages 4770–4779. IEEE.

Schwaber, K. 2004. *Agile project management with Scrum.* Microsoft Press.

Schwaber, K. and Beedle, M. 2002. gile software development with scrum.

Schwaber, K. and Sutherland, J. 2012. *Software in 30 days: how agile managers beat the odds, delight their customers, and leave competitors in the dust.* John Wiley & Sons.

Sidhu, J. S. and Volberda, H. W. 2011. Coordination of globally distributed teams: A co-evolution perspective on offshoring. *International Business Review,* 20(3):278–290.

Šmite, D., Moe, N. B., and Agerfalk, P. J. 2010a. *Agility across time and space: implementing agile methods in global software projects.* Springer Science& Business Media.

Šmite, D., Moe, N. B., and Agerfalk, P. J. 2010b. Fundamentals of agile distributed software development. In *Agility Across Time and Space,* pages 3–7. Springer.

Smith, J. L., Bohner, S., McCrickard, D. S. 2005. Toward introducing notification technology into distributed project teams. In *Engineering of Computer- Based Systems, 2005. ECBS'05. 12th IEEE International Conference and Workshops on the,* pages 349–356. IEEE.

Sutanto, J., Kankanhalli, A., and Tan, B. C. 2011. Deriving it-mediated task coordination portfolios for global virtual teams. Professional Communication, *IEEE Transactions on,* 54(2):133–151.

Sutherland, J., Schoonheim, G., and Rijk, M. 2009. Fully distributed scrum: Replicating local productivity and quality with offshore teams. In *System Sciences, 2009. HICSS'09. 42nd Hawaii International Conference on,* pages 1–8. IEEE.

Sutherland, J., Viktorov, A., Blount, J., and Puntikov, N. 2007. Distributed scrum: Agile project management with outsourced development teams. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on,* pages 274a–274a. IEEE.

Tripathi, N., Rodriguez, P., Ahmad, M. O., and Oivo, M. 2015. Scaling kanban for software development in a multisite organization: Challenges and potential solutions. In *Agile Processes, in Software Engineering, and Extreme Programming,* pages 178–190. Springer.