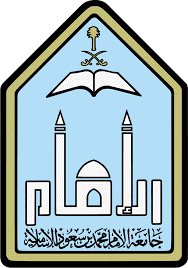
****

**Name :Jowaied Ali Salem Al marry**

University ID:437007150

Research Title : scrumban

Year: 1439H/1440H

**Dr.** **AbdulRahman Al Mubarak**

**Abstract**

Scrumban combines two Agile approaches (Scrum and Kanban) to create a management framework for improving software engineering practices. Scrumban is expected to override both Scrum and Kanban, as it inherits the best features of both. However, there is little understanding of the possible impact of Scrumban on software development in prior studies. This study first makes a comparison among Scrum, Kanban, and Scrumban and then investigates the impact of Scrumban on six major challenges of global software development. This study was conducted in a distributed project at two Software Factories in two universities in Finland and Italy. The results show that Scrumban could positively affect issues such as evenness of different sites, communication, and cultural issues as well as leveraging resources among sites. However, there are still few challenges that require alternative methodologies and tools other than Scrumban to be overcome.

**Keywords:** Agile, Distributed Software Development, Kanban, Scrum, Scrumban, Software Factory.

# 1.Introduction

The success of software development projects depends heavily on the use of appropriate software development methodology. According to a report by Standish Group, 42% of project cases which have used an Agile approach were successful, which is considerably more than what has been achieved using traditional project management methods [1]. Agile methods are iterative, incremental, and enhan ce collaboration between self-organizing cross-functional teams [2]. Scrum is the most frequently used Agile method in software development [3]. Scrum reaches its objective through time-boxed iterations based on continuous feedback and task prioritization [4].

Kanban, on the other hand, has not been widely adopted in software development [5].

In 2004, Kanban entered into the Agile realm when David Anderson introduced it in practice while assisting a software development team at Microsoft [6]. Kanban proposes to defer the project commitments, set constraints on the amount of work in progress (WIP), and limit the project promises that cause project failure [7]. The high expectations for Kanban are the result of its adaptability regarding changes in requirements, its ability to visualize project processes, and its role in increasing communication and cooperation among team members [8].

A study by Ladas [9] combined Scrum and Kanban to introduce a methodology which represents the best elements of those methodologies. According to that study, Scrumban is more appropriate for teams that are already using Scrum. Scrumban applies Scrum as a prescriptive method of team-work to complete the work, while it encourages process improvements through Kanban to allow projects to continuously optimize the processes and number of tasks [10].

# 2. Related works

This section discusses Scrum, Kanban, and Scrumban with respect to their similarities and differences. It also describes software development practices in a global context.

# 2.1 Scrum

The first implementation of Scrum in the field of software development was at Easel Corporation in the USA in 1993 [9]. Scrum advocates small teams that work independently and create more efficiency at work [10]. Scrum is an incremental Agile software development methodology that operates through a series of iterations that require continuous planning, defined roles, and project artefacts [2]. Scrum is the most frequently applied Agile software development method [3] for achieving small but continuous deliverables. It facilitates regular feedback after each iterative development process, called a ”sprint” [3]. According to Rising and Janoff [10], Scrum is beneficial, particularly for projects in which all the requirements are not clear in advance.

Implementation of Scrum allows self-organization which can result in a highperformance team even if the team comprises average developers [9]. The most important roles in Scrum include: (1) product owner, who serves as an interface among developers and other stakeholders, (2) Scrum master, who is the person responsible for leading scrum meetings, identifying tasks to be completed within the sprints, and measuring progress [20], and (3) development team. Since companies from Western countries often tend to outsource their software development to Eastern countries [1], applying Scrum in such situations can induce independency of teams as well as increase communication and productivity.

## 2.2 Kanban

Kanban is a relatively new concept in the field of software engineering that was originally applied in Lean manufacturing [6]. While Scrum focuses on one iteration at a time, Kanban supports a continuous workflow [5]. Kanban provides the flexibility of managing the workflow within teams. It limits 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 [4]. It provides a clear visualization of the phases in the project lifecycle.

Kanban reduces lead time and improves quality and productivity [5]. Kanban helps team members to identify constraints of a process and focus on a single item or task at a time [6]. In traditional software development methods, several works are assigned to a team member, which is defined as a push method [6]. In that case, the work to be completed is sent to the team member regardless of the status of other work. On the other hand, Kanban suggests assigning a developer to one particular job. When the work is completed, the developer can pull another task from the Kanban board and work on it. According to Polk [7], provision of a Kanban board changed the thinking of team members by making them realize that they are not just developing code but developing a complete product. With Kanban, team members, stakeholders, and customers can get a real-time view of project progress [6]. Implementation of Kanban also lowers the risk of communication and coordination breakdown [8].

# 2.3 Scrumban

By combining Lean and Agile methodologies, project members can receive rapid 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 colocated projects enhances coordination among team members, increases team morale, and produces better outcomes [3]. Lean increases the scale of the development process and makes it efficient, while Agile principles help to make the process flexible [1].

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 [9]. It is argued that if Kanban is used alongside Scrum, they can complement each other [3]. 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 [3]. By combining Scrum and Kanban, researchers hope to create more flexibility in projects as well as maintaining the iterative pace that Scrum has provided [3].

## 2.4 Distributed Software Development

Finding resources globally creates the possibility of mobility in resources and of accessing new knowledge of skilled people around the world [7]. Global software development is applied through multi-geo, multicultural, and multi-temporal environments to benefit from access to new markets, lower costs, increased operational efficiency, improved quality, and less time to markets [5].

DSD could have different configurational characteristics, which refers to the structural properties of the global environment, different ways of distributing developers, and differences in time and physical distance. A study by Ramasubbu et al. [6] examined how configurational dimensions can affect productivity, quality, and profit outcomes of distributed projects. This study explains aspects of dispersions including spatial dispersion to measure the physical distance, temporal dispersion to measure the time-zone difference, and configurational dispersion to measure structural properties such as number of distributed sites and homogeneity of distributed people and skills across different sites.

Smite et al. [7], Jim´enez et al. [7], and Nakamura et al. [8] declare thatˇ realizing the DSD benefits come with associated challenges 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.

# 3. RESEARCH PROCESS

## 3.1 Software Factory

Software Factories (SF) include structured sets of related software assets to provide developers with a development setting consisting of domain-specific tools that help to transform abstract models into implementations [6]. Through the SF 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 [7].

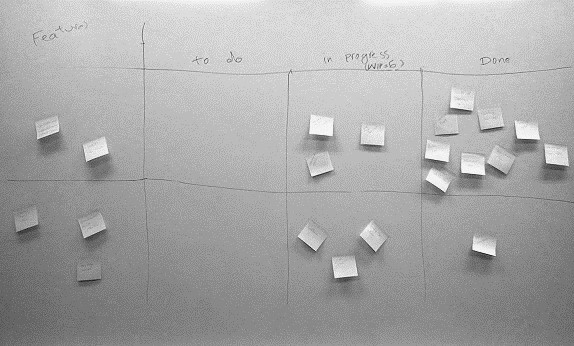
The SFs increase productivity from the business perspective, improve quality and consistency of architectures and designs, reduce development lifecycle, and consolidate operational efforts [7]. Also, SFs established in the context of universities are perfect avenues for exploiting technological research for innovation. A study by Taibi et al. [8] declares that such an SF environment benefits both business by receiving innovative, new ideas and academia by presenting new skills, frameworks, and models. Therefore, academic Software Factories can be additionally considered as a new concept of collaboration among universities and companies [8].

## 3.2 T-Bix Project Case

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 single common platform for the time-banking system to be operational in South Tyrol in Italy. The platform allows users to register their own profiles, search for jobs and products, post jobs and products, send requests for jobs and products, and communicate their feedback. The target group was young, unemployed people as well as senior citizens hit by the socio-economic crisis.

Because T-Bix project teams were located in Europe (North-South dispersion configuration), they did not experience drastic temporal differences; however, the long physical distance and diverse cultures, languages, and social behaviors remained as challenges in the project.

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

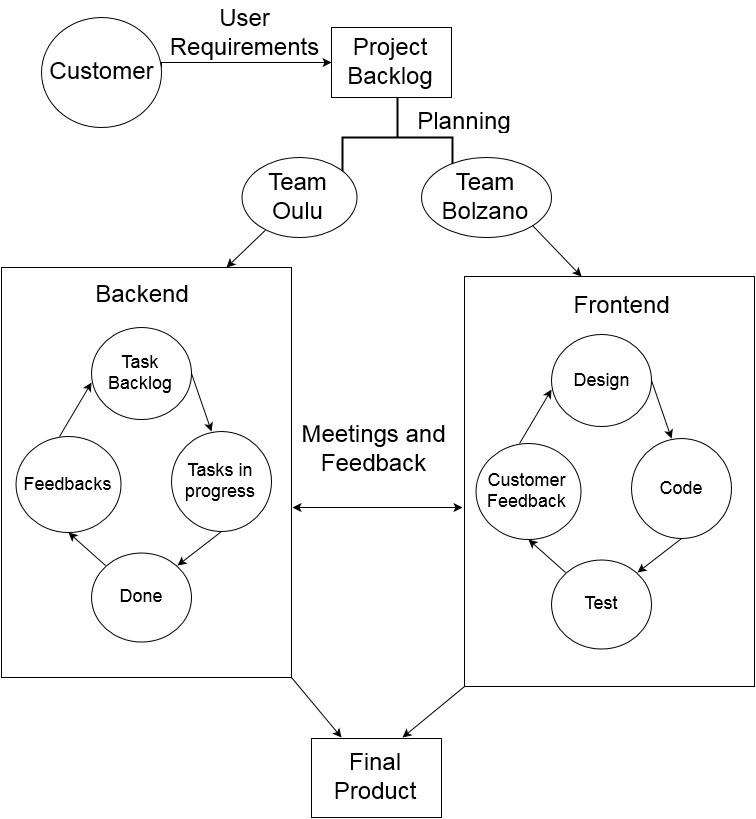


**Fig.1.** A physical Kanban Board in Oulu Software Factory

## 3.3 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.

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 latest work. 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 the project was carried out among the teams.



**Fig.2.** Project coordination model

## 3.4 Research Approach

This study exploits empirical software engineering methods. The authors have applied semi-structured interviews to collect the data. The participants of this study are members of the Oulu Software Factory who were interviewed after completion of the project. The authors provided a set of open-ended questions covering the scope and objectives of this paper. Four rounds of interviews were conducted, which lasted from 45 minutes to 2 hours. All interviews were recorded and transcribed, enabling authors to 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 semistructured interviews is regarded as reliable for analysis [9].

# CONCLUSIONS

Current software companies tend to establish their production units in different locations in order to optimize skilled workforces to produce products at higher quality and lower cost. In that regard, companies need adequate methodologies, techniques, and tools to improve efficiency and decrease challenges in DSD. Software Factory settings can reuse existing assets, architectures, knowledge, and components to develop software artefacts by imitating industrial processes. The current study has used Software Factories to investigate how coordination among distributed sites is effected by the combination of Scrum and Kanban. Our research shows that the full extent of Scrumban capability is still unknown because it has not been researched a great deal. Therefore, the results of this research can be used as the initial steps for developing and validating an efficient methodology for software engineering practices, particularly in distributed sites. There are different issues which should be considered before companies decide to locate their branches in various remote sites. This study argues that Scrumban could alleviate some of those challenges, but further solutions are needed to make DSD more reasonable than co-located developments. Furthermore, companies must restructure their organizations to include proper roles and processes to improve transparency, change management, communication, coordination, and resources in DSD.

Future scholarly studies could investigate perspectives other than coordination for the usability of Scrumban. Moreover, they could propose new domain-specific tools and approaches for DSD projects which impose different constraints, for example, East-West distributed projects. However, companies, as the real users of Scrumban methodology, should be aware of its challenges as well as its benefits in planning project deliverables and coordination among teams.

# References

1. Schwaber, K. and Sutherland, J.: Software in 30 days: how agile managers beat the odds, delight their customers, and leave competitors in the dust. John Wiley & Sons Press (2012)
2. Alam, S. S. and Chandra, S.: Agile software development: Novel approaches for software engineering. International Journal Of Engineering And Science (IJES). 3(1), 36-40 (2014)
3. Rodriguez, P., Markkula, J., Oivo, M., and Turula, K.: Survey on agile and lean usage in Finnish software industry. In Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement, pp. 139-148. ACM Press (2012)
4. Yilmaz, M. and O’Connor, R.: A Scrumban integrated gamification approach to guide software process improvement: a Turkish case study.Tehnicki Vjesnik. 23(1), 237-245 (2016)
5. Mahnic, V.: Improving software development through combination of scrum and kanban. Recent Advances in Computer Engineering, Communications and Information Technology, Spain (2014)
6. Ahmad, M. O., Markkula, J., and Oivo, M.: Kanban in software development: A systematic literature review. In 39th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), pp. 9-16. IEEE Press (2013)
7. Reddy, A.: The Scrumban [r] evolution: Getting the Most Out of Agile, Scrum, and Lean Kanban, Addison-Wesley Professional (2015)
8. Kniberg, H. and Skarin, M.: Kanban and Scrummaking the most of both. The InfoQ Enterprise Software Development (2010)
9. Ladas, C.: Scrumban. Lean Software Engineering-Essays on the Continuous Delivery of High Quality Information Systems (2008)
10. Khan, Z.: Scrumban-adaptive agile development process: Using scrumban to improve software development process. Master’s Thesis, Finland (2014)