

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/341318650>

Framework Study for Agile Software Development Via Scrum and Kanban

Article in *International Journal of Innovation and Technology Management* · May 2020

DOI: 10.1142/S0219877020300025

CITATIONS

23

READS

3,741

2 authors:



Wael Zayat

Marmara University

3 PUBLICATIONS 31 CITATIONS

[SEE PROFILE](#)



Ozlem Senvar

Institut Charles Delaunay, Université de Technologie de Troyes

42 PUBLICATIONS 626 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Computation [View project](#)



lean six sigma project [View project](#)

Framework Study for Agile Software Development Via Scrum and Kanban*

Wael Zayat[†] and Ozlem Senvar[‡]

*Department of Industrial Engineering
 Marmara University, Istanbul, Turkey
[†]wael.zayat88@gmail.com*

[‡]ozlem.senvar@marmara.edu.tr; ozlemsenvar1@gmail.com

Received 24 March 2019

Accepted 7 May 2020

Published 24 July 2020

This paper provides a systematic comparison between two well-known Agile methodologies: *Scrum*, which is a framework of doing projects by allocating tasks into small stages called sprints, and *Kanban*, which is a scheduling system to manage the flow of work by means of visual signals. In this regard, both methodologies were reviewed to explore similarities and differences between them. Then, a focus group survey was performed to specify the preferable methodology for product development according to various parameters in the project environment including project complexity, level of uncertainty, and work size with consideration of output factors like quality, productivity, and delivery. Results show the flexibility of both methodologies in approaching Agile objectives, where Scrum emphasizes on the corporation of the customer and development teams with a focus on particular skills such as planning, organization, presentation, and reviewing which makes it ideal for new and complex projects where a regular involvement of the customer is required, whereas Kanban is more operative in continuous-flow environments with a steady approach toward a system improvement.

Keywords: Agile software development; Scrum; Kanban; software development life cycle, project management.

1. Introduction

Challenges in innovation processes have guided the development of Agile methodologies for both project management and software development [Tura *et al.* (2017)]. Software development methods are continuously evolving along with the new challenges of today's markets. Traditionally, software development was guided by life cycle models such as the waterfall model. In a life cycle model, tasks are specified, and responsibilities are assigned to individuals with different roles such as a project manager or an analyst. The problem with these methodologies is the huge amount of paperwork needed for communication among project participants since everything

*The review of this paper was managed by IJITM Associate Editor Alexander Brem.

[‡]Corresponding author.

must be formalized through documents. The involvement of the customers in this case is usually limited to the first stage of developing the specifications for the requested product [Nerur *et al.* (2005)].

The Waterfall approach to system development is a well-known process that was originally defined by Royce [1970]. It quickly gained support because everything flows smoothly from the beginning of a project until the end (as shown in Fig. 1). The detailed procedures can differ in the process depending on the application; however, the basic steps are the same.

The model of waterfall defines the fundamental activities of building the requirements of a product, followed by a validation process. This process is done through a set of separate stages such as requirements' analysis, design, implementation, testing, and maintenance [Birgün and Çerkezoğlu (2019)]. This model works well where requirements (and risks) are already set in advance with a very limited amount of modifications to be expected during the development of products. Otherwise, if managers, users, and developers are still experimenting the features they want during the phases of development, the waterfall model can go through many problems [Smith (2005); Cockburn (2006); Davis (1982)]. Agile development is introduced to properly handle such a problem [Beck *et al.* (2001); Conboy (2009)]. Agile software development is a set of methodologies where solutions are developed by the collaboration of different teams with different experiences. The coordination between customers and software engineering team leads to a high level of adaptation with any additional requirements and encourages quick responsiveness to customer orders. An important feature of Agile is its support for the culture of change with the idea that change will have a positive impact on the development of products. Furthermore, it embraces the idea of following a clear model to react and learn from changes. Agile was the natural response to the demand of today's markets which expects innovative and high-quality software products in a very short period. Agile software development satisfies two of the most critical needs for most of today's business and technology workplaces: the innovative, dynamic approach to run a project, and the desire to build workplaces in a creative way that guarantees an effective communication between people. The active communication of developers, customers, and sponsors enables them to overcome difficulties, set priorities, and investigate alternate paths. This can save costs, speed up the process, and achieves maneuverability [Highsmith and Cockburn (2001)].

One of the most popular methodologies based on Agile development is known as Scrum. Scrum is a project management approach that works according to iterative and incremental developments. The stages of Scrum model start with product backlog creation as the first stage. Afterward, the second stage is processed through sprint planning and sprint backlog creation. Each sprint starts with a sprint



Fig. 1. Waterfall model [Birgün and Çerkezoğlu (2019)].

planning where Scrum team gathers in a long meeting (could take 4h) to plan the sprint in detail. The third stage includes working on sprints and meetings. Then, the software will be developed as increments in the sprints where each sprint is usually set up to two to four weeks. Usually, 15-min meeting is set up every day with three main questions: what has been done the day before, what will be done today, and what immediate challenges are to be faced. In the fourth stage, a performance testing after each sprint is performed. Finally, in the fifth stage, retrospective and future sprint planning are accomplished. In this stage, stakeholders attend a review meeting to assess the teamwork in the sprint. Scrum backlog shows customer requirements and daily burn-down charts assess the remaining work [Hossain *et al.* (2009)].

Another well-known methodology is Kanban which is developed by Toyota production system as a part of Lean manufacturing [Ohno (1988); Womack *et al.* (1990)]. The main advantages of Kanban are improving lead time needed to deliver the product, improving the quality of the product, emphasizing the coordination and communication among team members, achieving consistent delivery of products, and minimizing customer defects [Ahmad *et al.* (2013)]. Nowadays, Scrum and Kanban methodologies are the most adopted methods by system development organizations around the world.

Applicants of Scrum or Kanban as Agile methodologies within the software development sector must overcome several challenges. Bjørni and Haugen [2019] revised several papers that concluded key challenges in Agile adaptations with a particular focus on Scrum application. Then a case study was presented to discover the causes and consequences of those challenges. These challenges include the proper estimation of sprint workload, testing, documentation, team improvement, unclear backlog items, agility, team improvement obstacles, product release, and sprint review challenges. The authors also gave some guidelines to overcome those challenges. Ahmad *et al.* [2018] revised Kanban applications in software development. The authors highlighted the main benefits and challenges of applying Kanban. Challenges face Kanban practitioners were divided into three groups: process, people, and organizational challenges. Process challenges covered Kanban understanding and implementation challenges along with the challenges in assessing the performance with the appropriate metrics. People challenges dealt with the motivation of new practices and the ability to manage unexpected development of workflow. Organizational challenges involved overcoming training issues and cultural issues. This study argues that it is not enough to overcome those challenges for the successful application of Scrum and Kanban but rather to understand the work environment itself. Hence, this study aims to highlight similarities and differences between Kanban and Scrum and to determine the proper methodology to apply in different work environments. The work of Lei *et al.* [2017] also meant to compare the two methodologies. It provided a statistical analysis of the effects of Scrum and Kanban on software development projects. The authors gave their analysis based on a survey where 35 people (60% used Scrum and 40% used Kanban) were asked to answer a set of 14 questions that reflected six main factors including risk, quality, resources, schedule, scope, and budget. Their results showed that Kanban was

superior over Scrum in its scheduling system, scope of time needed for projects, quality of products, handling of resources, and managing risks. On the other hand, Scrum was superior in managing budget and the return of the investment. The work of [Stoica et al. \[2016\]](#) analyzed Agile development from waterfall style to Scrumban. The work classified software development using three-level pyramid where Agile and Lean were considered as philosophies, Scrum and Kanban as methodologies, and sprints, boards, charts, etc. as tools. Also, a comparison between the main aspects of Scrum and Kanban in general. [Yordanova and Toshkov \[2019\]](#) compared between Scrum and Kanban with an attempt to create a new Agile methodology that uses tools and techniques from both. [Mircea \[2019\]](#) also made a comparison between the two methodologies and extended the comparison to include Scrumban. The author's work excluded key concepts that should be built to ensure the success of Agile teams in both methodologies. [Kumar et al. \[2019\]](#) studied 13 different Agile methodologies as well. The authors provided a comparison among those methodologies based on 50 different parameters of projects. [Emelyanova et al. \[2020\]](#) provided advantages and disadvantages of nine software development methodologies including waterfall model, Scrum, and Kanban. According to their study, 58% of Agile applications used Scrum, whereas only 10% used Kanban for their applications. However, in this paper, the success of application of Kanban or Scrum is highly dependent on some criteria regarding work environment and the similarities and differences are discussed in more detail. In this regard, a focus group survey is performed where a group of professionals in Agile software development and Agile project management was asked about their experiences in applying Scrum and Kanban.

The rest of the paper is organized as follows: In Sec. 2, Agile development methodologies are explained. In Sec. 3, similarities between Kanban and Scrum are provided. In Sec. 4, differences between Kanban and Scrum are given. In Sec. 5, the research methodology is summarized. Results and discussion are presented in Sec. 6. Section 7 gives conclusion along with recommendations.

2. Agile Development Methodologies

In early 2001, 17 software development practitioners had a meeting in Snowbird, Utah to share various ideas about different methods of software development and they came up with the idea of Agile [[Koch \(2005\)](#)]. A manifesto for Agile software development was created with a set of values and corresponding 12 principles [[Black and Coleman \(2017\)](#)]. Agile was exactly what developers needed to deal with more varied requirements of customers. In addition, Agile does not require a documentation system for the product; it rather focuses on the customer in a way that builds up confidence between the developing team and the customer. This may create the main drawback of Agile model.

Figure 2 shows system (or software) development life cycle (SDLC). In this cycle, each stage of the development process includes a set of activities that require certain tools. Moreover, a detailed plan is required to illustrate the methods for

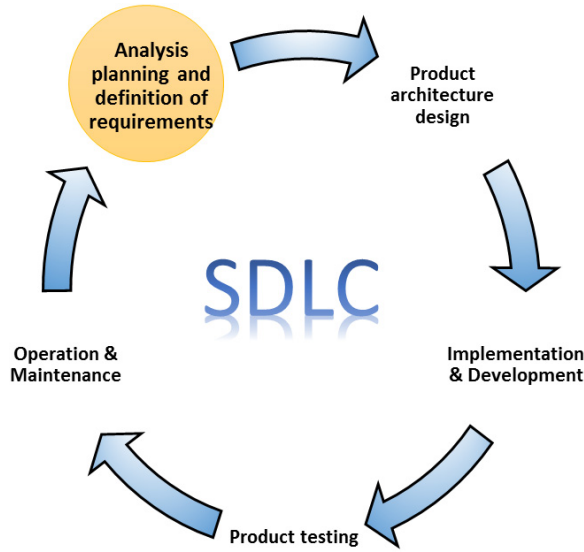


Fig. 2. System development life cycle (adapted from [Stoica et al. \[2016\]](#)).

development, maintenance, and validation of the product according to the standards of ISO-IEC 12207.

The international standard of ISO-IEC 12207 categorizes the processes to be performed in SDLC to seven different groups. This standard illustrates the purpose and desired result of each group along with the activities and tasks required to achieve the desired outcome. For this paper, we focus on two main life cycle processes which are the project processes and the software implementation processes. The project processes are classified as project management processes, which is performed to execute, assess, and control the development of project, and project support processes, which is performed to support the management objectives by assessing the performance and managing the information and risks of the project. The processes of software implementation start with the requirements analysis. Then a software architectural design is developed, followed by detailed design. After that, the construction process of the software takes place. Later, a software integration process is performed, and finally, a qualification test is done to check whether the software meets the requirements of the customers. ISO-IEC 12207 does not require a specific model of application; however, the tasks of each process should be met [[Singh \(1996\)](#)].

To reach the objectives of a project, several processes and methodologies are used in the activities performed along the cycle of SDLC. This again demonstrates how optimal solution can be reached using different methods based on the situation [[Stoica et al. \(2016\)](#)]. Agile constructs a robust platform to meet the requirements of ISO-IEC 12207. Perhaps, the biggest difference between Agile and other traditional methods is that Agile projects are developed incrementally. Figure 3 illustrates the increment process of Agile [[Stoica et al. \(2016\)](#)].

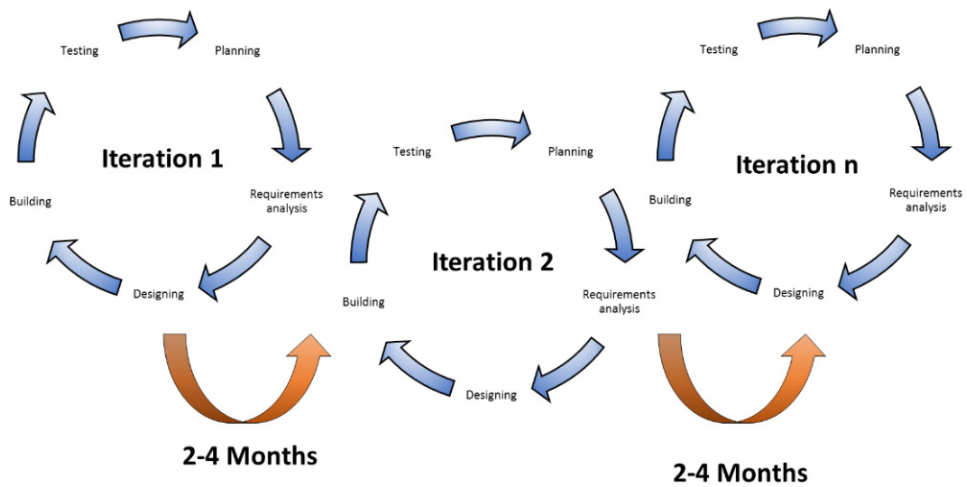


Fig. 3. Agile model diagram (adapted from Stoica *et al.* [2016]).

As seen from Fig. 3, the incremental development of SDLC is done through n number of iterations where a quality test is performed in each increment. Incremental development is attained when each successive version of the product builds up the previous version by including a new subset of features. Each increment is dealt with as a potential shipment; consequently, it is fully completed and tested. In Agile systems, this development is iterative since customer requirements cannot be clear at the beginning of a project; therefore, new feedbacks are proposed in each iteration. This iterative evaluation allows for modifications to the initial design before releasing the product while users' feedbacks are collected after the deployment stage. These iterations guarantee a better quality of products since more communication is established. All of these occur according to a certain time frame. The number of iterations n depends on user stories in the product backlog. At the beginning of each project, all the stories are being discussed and assessed according to complexity, risk, number of unknowns, and efforts. Story points are assigned to each story based on estimated size. Then the workload needed for all the stories, which is the total estimated points, is assigned to n number of iterations. Burn-down chart is plotted at this step (see Fig. 4). The number of story points delivered is presented on a chart to show the remaining workload. This can provide a visual representation of the work that is yet to be processed. The speed of performing tasks will be clear when progress is entered over time. This can give an estimation of project completion time. Velocity can be measured in story points and it is usually improved as the development team becomes more familiar with the tasks until it reaches a constant level [Winter (2019)].

Agile is built upon 12 core principles that shape the structure of Agile framework [Martin and Martin (2006)]:

- Satisfying the customer is the management highest priority which is done through the early and continuous delivery of valuable products.

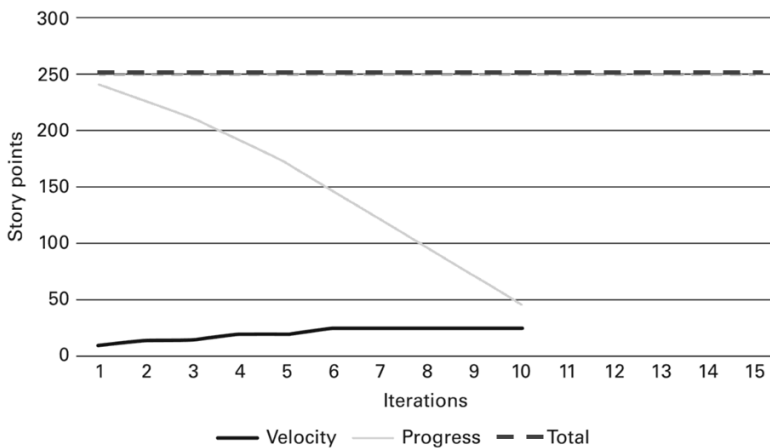


Fig. 4. Burn-down chart [Winter (2019)].

- Changing in requirements is welcomed even if it came late in development.
- Products are delivered frequently, from a couple of weeks to a couple of months with a shorter timescale in preference.
- Developers must work with business managers together on a daily basis throughout the project.
- Projects are being built by motivated people. They are given the support and environment they need, and they are trusted to do the work.
- Face-to-face conversation is the best and most efficient way of delivering information to the development team.
- The primary measure of progress is done by working software.
- Processes in Agile are developed continuously. A constant pace should be maintained by sponsors, developers, and users.
- Agility is enhanced through persistence focus toward good designs and technical excellence.
- Simplicity is very important in the mean of maximizing the work that is not done.
- Best architectures, requirements, and designs arise from self-organizing teams.
- After regular intervals, teams can understand their weaknesses and can overcome obstacles to become more efficient.

Agile intention is to guarantee an effective collaboration between the customer and the development team by encouraging the customer to engage in every step of the project starting by prioritizing the user stories to iteration planning and iteration reviews. This reflects a high level of transparency between the customer and the developing team. Agile focuses on business values because as the customer engages in the prioritization of the features, teams can understand what is important for the customer's business. Furthermore, using a time-boxed schedule for product delivery allows the cost to be predictable and limited to the amount of work that can be performed in a number of iterations. The customer can understand the cost of each feature according to the time it takes which improves decision making about the

priority of features and the need for additional iterations. Quality improvement is another important intention of Agile software development. The project team can focus on high-quality development and testing by breaking down the project into small parts. During an iteration, as the team is focused on developing the features that have been agreed upon, the customer can reprioritize the product backlog to introduce a new or modified item for the next iteration which provides flexibility to deal with changes. Providing a potential product at the end of each iteration can also provide an opportunity for a beta test. This can provide valuable feedback allowing to make changes as needed. Also, fixing defects quickly during the testing phase at each iteration and receiving the customer feedback ensure a high-quality product with high customer satisfaction. Therefore, Agile has grown widely in the market of software industry. Scrum and Kanban are two of the most popular Agile methodologies that have proven to be effective with thousands of companies from the beginning of this century. Building on Agile principals, Kanban and Scrum can facilitate the way to achieve Agile goals.

2.1. *Scrum*

Scrum is defined as a framework in which people can deal with complex problems, while maintaining high productivity in delivering high-quality products [Schwaber and Sutherland (2017)]. Scrum is part of Agile software development. The word Scrum came from rugby sport, where Scrum is the formation where a quick adoption of strategies is built by team members with every player plays a specific role.

2.1.1. *Evolution of scrum*

The origin of Scrum goes back to 1995 when Jeff Sutherland and Ken Sutherland presented the process in the object-oriented programming, systems, languages and applications (OOPSLA) conference in Austin, Texas. Afterward, Ken Sutherland published a paper demonstrating the methodology [Schwaber (1995)]. The whole idea was based on another paper where the word Scrum was used due to its relation to the sport of rugby in order to highlight the importance of collaboration between team members for successful projects. Sutherland's work presented a new idea that is very effective to use in complex projects. The idea is to develop organized teams with a set of objectives rather than tasks. Later on, the very first book on Scrum was published [Schwaber and Beedle (2001)]. In 2002, Scrum alliance was founded, after that Certified Scrum-Master programs were given all around the world.

2.1.2. *Scrum team*

In ever Scrum team, there is a product owner, a Scrum master, and the development team. Teams in Scrum are cross-functional and self-organized, which means that they choose how to perform instead of being managed by others. Teams are designed to be flexible and to work with high creativity and productivity. Products are delivered iteratively which creates opportunities for customer feedback at each increment.

2.1.2.1. Product owner

The main job of the product owner is to make sure that the product is of high quality. This is done in various ways from an organization to another. Usually, product owner is one person, and he is responsible for managing the backlogs of the product which includes the following:

- Identifying backlog items.
- Putting product items in an order that best suits the goals of the project.
- Improving the quality of work done by the development team.
- Clarifying the backlog for all team members.
- Making sure that everyone knows how to proceed.

It is important for backlog items to be directed by one person even if the priority of those items is determined by a committee, they should address the product owner to make the necessary adjustments. For the Scrum team to be successful, everyone must respect the decisions made by the product owner which are reflected in the product backlog.

2.1.2.2. Scrum master

He is the one responsible for supporting the application of Scrum as instructed in the Scrum guide. Consequently, he makes sure everyone understands Scrum theory, application, rules, and values. Scrum master shows those who are new to Scrum how to interact with the team in a way that ensures the maximized value created by Scrum team. The Scrum master should work only on one project at the time to give this project full attention. Although Scrum master does not have a managing role, he should try his best to increase the effectiveness of the project.

2.1.2.3. Scrum development team

At the end of each sprint, the new product is delivered as an increment by the development team. This team includes professionals who are responsible for building the project according to the desired specifications in the product backlog. The optimization of team structure will result in the effectiveness of the team.

The following characteristics should exist in the development team:

- Development team is self-organized. This means that no one (including the Scrum master) can tell them how to turn backlogs into a releasable product.
- They are cross-functional; they do not need anyone from outside to complete the work.
- No titles are being given to any of the members no matter what work they perform.
- In Scrum framework, there are no sub-teams even though most of the time, members work in different domains such as operations, architecture, and testing.
- Members have specialized skills; yet they can be flexible to work in different areas within the team.

2.1.3. Scrum framework

Everything starts with organizing the product backlog, which is processed under the supervision of the Scrum master. Then, the sprint planning starts, followed by the sprint, which results in a deliverable product ready for the customer. After that, the same cycle repeats iteratively. Figure 5 illustrates Scrum framework which includes these stages:

- **Product Backlog:** The product owner lists user stories items and prioritizes them in the right order. This is done by determining the important items or features that must be on the product, followed by items that are less important and finally the items that are not fit in the timeframe. This will ensure the clarity of the project elements and is done based on the impact, risk of items, and the way they help in the learning process of the future product later in the project [Kniberg and Skarin (2010)].
- **Sprint Planning:** Development team starts working with backlog items based on their priority. Team members first decide how to build it. The development team should not care much about the detailed level of developing items at this level, and they should think about the items that are ready to be worked on. In this stage, development team is encouraged to ask product owner and stakeholder questions about the required features in the product [Kniberg and Skarin (2010)].
- **The Sprint:** In this stage, the developing team will work on the user stories that were agreed upon for the next shipment of the product. These stories are chosen from the product backlog during the planning meeting according to priority and the estimated work needed, which should fit within one sprint. The sprint starts with a daily meeting to make sure that everything is going as planned. During a sprint, the sprint backlog is not allowed to change except for certain blocks of unresolved issues. A sprint is considered as a project which has certain objective and scope where all the features that are added at this stage are completed, tested, and ready to be delivered as the new product increment. Usually, sprints take short duration of two to four weeks, but not more than one month. This duration

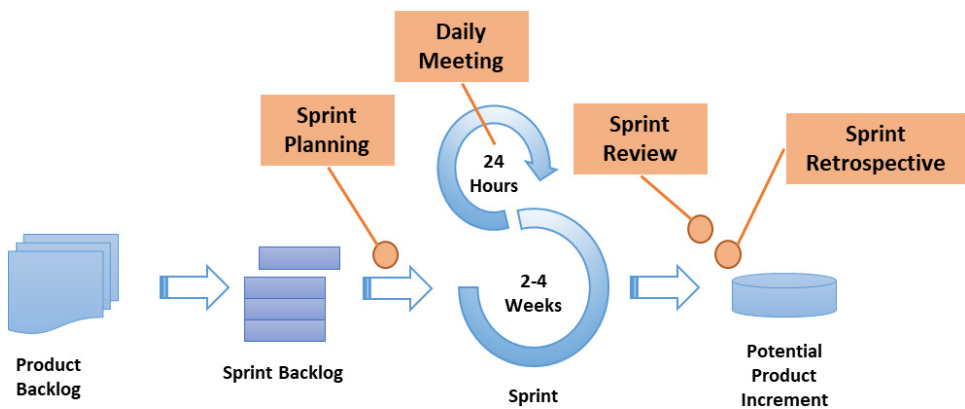


Fig. 5. Scrum life cycle (adapted from [Tavares et al. (2019)]).

can be revised with the coordination of the product owner as the team learns more about the project. When a sprint's duration is long, the product backlog may change, which could add more risks, costs, and complexity to the project [Schwaber and Sutherland (2017)].

- **Completed Sprint:** When the work is done, and the project is ready for delivery, a sprint can be completed. At this point, some tests can be performed by the customer or the shareholders. Automation can play a role here since it can help solving the issues of cost and time [Kniberg and Skarin (2010)].
- **Review:** After the sprint is over, a meeting is set to review what worked well and what did not. This is very important for future sprints. Usually, Scrum team tries to maintain the positive implementations and remove any obstacles. A discussion should have some ground rules to maintain positive energy along the meeting [Kniberg and Skarin (2010)].
- **Repeat:** The same cycle repeats over and over starting from items at the top of the product backlog following the same steps and improving the process through experiencing new ideas and refining work to reach a high quality of performance [Kniberg and Skarin (2010)].

Scrum can only be applied when teams are qualified. A typical problem in Scrum applications is its underestimation of deadlines and budget [Emelyanova *et al.* (2020)]. Scrum performance can be assessed by team velocity chart and burn out charts. Dixit and Bhushan [2019] demonstrated these two charts and designed several metrics to measure the accuracy of sprint planning as well as the software defects along Scrum applications.

2.2. Kanban

Kanban, which means visual signal, was first used by workers in Toyota to track processes on their manufacturing system. Using Kanban, teams were able to communicate more effectively as this simple tool provided instance information on what needed to be worked on and when it is needed [Lei *et al.* (2017)]. The concept of Kanban is one of the fundamental tools of just in time (JIT) and Lean manufacturing. Kanban is a scheduling system that provides information about what is needed, when to deliver, and how much to is needed [Ohno (1982)]. The entire value chain is controlled ideally by Kanban starting from the supplier to the customer.

Kanban system avoids supply disorder and eliminates overstocking (caused by traditional push systems) of products across different levels of the manufacturing system. One of the biggest advantages of working with Kanban is that it is easy to spot bottlenecks that can affect the flow of work across the workstations. Kanban also measures productivity by means of lead time. Overtime, Kanban proved its efficiency in various workplaces, and it is now one of the most popular production methodologies across the world.

Although Kanban is designed to eliminate waste by minimizing non-value-added steps, various kinds of wastes are likely to appear and should be handled separately. Lean thinking encourages continuous improvement throughout understanding the

source of each kind of waste and eliminating this waste properly [Ikonen *et al.* (2010)].

2.2.1. *Evolution of Kanban*

Kanban methodology goes back to the late 1940s when Taiichi Ohno developed the system for Toyota in Japan based on the same model that supermarkets were using in stocking their shelves. The idea is that sales men did not fill up their stocks based on their vendor's supply, but rather according to their needs [Ohno (1982)]. In 2004, David J. Anderson implemented the concept of Kanban to software development and information technology systems in general. Anderson built up the ideas of Taiichi Ohno, Edward Demmings, Eli Goldratt, and others to introduce Kanban with methods of pull, flow, and queuing theories for knowledge work [Anderson (2010)]. In today's Agile systems, Kanban is a very popular framework. In fact, the reports of annual state of Agile show that in 2015, Kanban users were increased from 31% to 39% and in 2016 from 39% to 50% [One (2016, 2017)].

Various lean principles are demonstrated in Kanban. Perhaps, the greatest attribute of Kanban is that it provides a clear visualization of the assigned work of each developer; also, it communicates priorities and minimizes work in process (WIP) by developing only requested items which leads to continuous flow of items. The main focus of Kanban is the flow of work along with the elimination of unnecessary activities which leads to shorter feedback loops. Lean software development principles include eliminating waste, improving quality, creating knowledge, fast delivery, and optimization. These principles are reflected in Kanban through visualizing workflow, limiting WIP, measuring and managing the flow, clarifying process policy, improving collaboratively (using models and the scientific method), and productivity improvement. It is clearly established how Kanban principles overlap with principles of Lean Software development [Ahmad *et al.* (2013)].

2.2.2. *Kanban framework*

- **Visualize Work:** In order to start working with Kanban, a board is divided into columns that represent stations in the workstream and then cards will move from a column to another simulating what is really happening in the actual system. By visualizing the work in this way, every move can be observed, tracked, and evaluated easily.
- **WIP:** This step is done by minimizing the size of work to be done before the task is delivered; this minimizes the time needed for an item to travel across the board of Kanban.
- **Focus on the Flow:** Kanban system can be optimized by analyzing the flow of work through the processes. Some basic principles can be followed to limit the work in process and to avoid any bottlenecks in the system.
- **Continuous Improvement:** Great systems always have a mechanism of reaching a better state of performance through time. This mechanism is processed

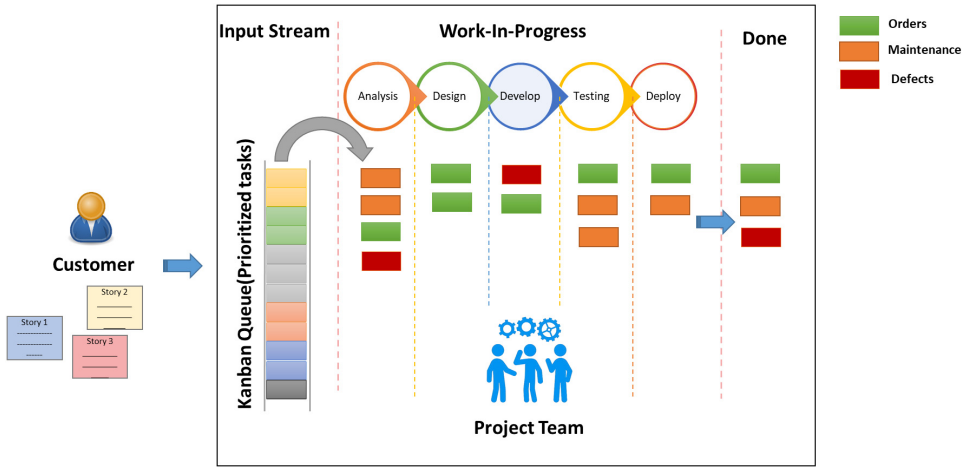


Fig. 6. Kanban board (adapted from Kumar *et al.* [2019]).

in Kanban through eliminating problems within the system with an objective to reach a shorter lead time, a better quality, and a more coherent flow.

2.2.3. Kanban board

Basic Kanban board includes the following three main columns:

- To do: all the tasks that are yet to be started are listed in this column. Generally, tasks are sorted according to their arrival times; however, sometimes, this is done according to priorities.
- In progress: this column is allocated to the tasks that are being worked on at the moment.
- Done: which shows the completed tasks.

This simple approach can lead to great clearness about what should be done, and it shows any existing bottlenecks if any. Sure, the board can be extended to include as many columns as needed for the framework, as shown in Fig. 6.

3. Similarities of Scrum and Kanban

After a deep understanding of the two methodologies, many similarities are observed including:

- Both methodologies make profound platforms of the Agile 12 principals explained in Agile manifesto.
- Both break up projects into small tasks presented by cards in Kanban system, and by the mean of user stories in Scrum [Yordanova and Toshkov (2019)].
- Development teams are self-organized and well managed.
- Both are pull systems that work according to customer orders.

- Transparency is well established in Kanban by the visualization of cards on the board and in Scrum through the daily face-to-face meeting or yet again using the board [Yordanova and Toshkov (2019)].
- Both place a high value on continual improvement, optimization of the work, and the process.

4. Differences Between Scrum and Kanban

- **Roles:** In Scrum, every team has a well-defined role to play. The project owner defines what to do, the Scrum master controls the time progress, and development teams do the work, whereas in Kanban there may be a project manager to control the whole process [Yordanova and Toshkov (2019); Mircea (2019)].
- **Metrics:** In Scrum, time is measured using the velocity of a sprint; in Kanban, the time is measured using lead time. Both are proper measures for planning and improvement.
- **Flexibility to Change:** Scrum system is fixed during sprints. Once the sprint is started, it is highly discouraged to add any tasks to the sprint. However, changes can be made during the planning meeting. Kanban on the other hand is very fixable and can take many shapes according to different applications [Yordanova and Toshkov (2019)]. According to Kumar *et al.* [2019], Kanban can be exposed to frequent changes in the specifications which can overperform Scrum at this point which can only deal with medium level changes.
- **User Participation:** Scrum encourages user participation and involvement in the development process much more than Kanban. Although applying Kanban as an Agile methodology implies that face-to-face meetings with customers are arranged, these meetings are not as frequent as in Scrum [Kumar *et al.* (2019)].
- **Elasticity:** Scrum is more rigid than Kanban as a system. Boards, operations, and delivery time can be more flexible [Kumar *et al.* (2019)].
- **Meetings:** Scrum encourages face-to-face interaction, starts the sprint with a planning meeting, follows the progress of the work on daily basis, and finalizes by having a sprint review. In Kanban, everyone starts directly. Project members can follow some meeting schedule for improvement, but since it is a more continuous process in Kanban, usually everyone knows exactly what to do and how to proceed. [Yordanova and Toshkov (2019); Mircea (2019)]
- **WIP:** Scrum starts by defining all user stories that will be processed during the sprint; WIP is minimized per each sprint. Kanban uses the capacity of each workstation and manages the flow of cards accordingly with as limited WIP as possible [Yordanova and Toshkov (2019)]
- **Teams:** Scrum teams are cross-functional; they need to interact with each other, but each has his own tools. In Kanban, all teams use the same board and every team is specialized in a specific task [Yordanova and Toshkov (2019); Mircea (2019)].

- **Board Status:** Scrum reset the board (if used) after each sprint (two to three weeks), while the board is always persistent in Kanban.
- **Delivery:** Scrum system is designed to deliver products after the review at the end of each sprint. In Kanban, the delivery is a continuous process [Yordanova and Toshkov (2019)].

5. Research Methodology

In this study, focus group methodology, which is a guided qualitative research technique, is performed in order to ask professionals for open-ended responses conveying their personal experiences and preferences regarding the implementation of Scrum and Kanban. As prescribed in the literature, a focus group should be about 6–10 people who interact with each other to get better results. However, in today's world, the internet made it possible for as much as needed people to be involved in such an assembly. For this research, the group involved more than 40 practitioners who are experts in working with Agile systems. Focus group technique can be more valued than traditional surveys when the discussed questions require experience in the subject of the study, which is the case here.

6. Results and Discussion

Responses given by practitioners provided important aspects that guide new Agile applicants to choose the methodology that works best according to different workplace parameters. Furthermore, application outcomes were considered for a better understanding of key features in Scrum and Kanban.

6.1. Workplace parameters

Both models have their advantages and disadvantages depending on different parameters of the project. In order to choose the right method, the following aspects must be determined.

6.1.1. Environment of work

Scrum is preferred with project-oriented environments where there is a certain objective to be reached at the end of the project. In this case, every sprint will be a milestone in achieving the final objective of the project. Kanban on the other hand is preferred with continuous-flow environments where work needs to be done without an expected end such as assembly lines productions and repetitive activities task. Figure 7 illustrates the difference according to the environment of work.

6.1.2. Frequency of change in orders

When teams are supposed to respond quickly to new issues, or when dealing with projects that are exposed to many changes, or when there is uncertainty in the priorities of the project, Kanban is more suitable to apply. Change in Scrum is very

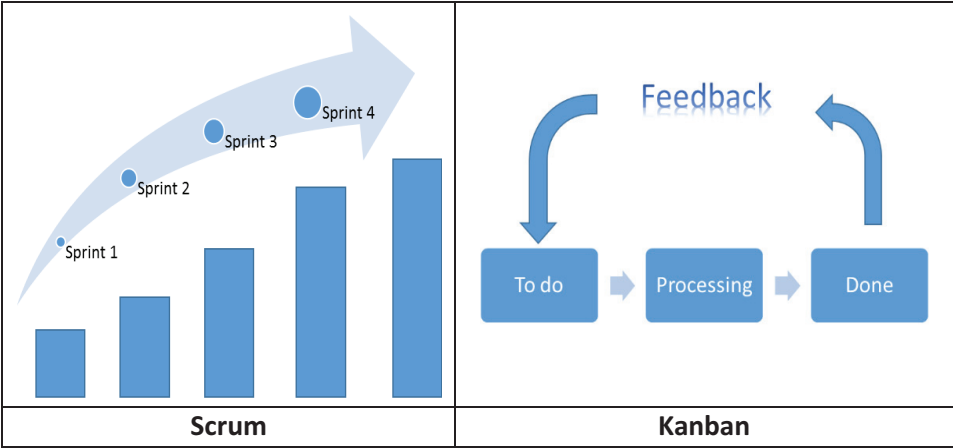


Fig. 7. Scrum and Kanban differences according to the environment of work.

hard after the sprint is already in process. However, Scrum can be more effective than Kanban when a block of work is to be achieved within a cohesive date. Figure 8 represents the proper system based on the change frequency of the project.

6.1.3. Level of complexity

Scrum can be a powerful methodology for organizing complex teams and tasks that interact with each other. This results from many features embedded in Scrum including planning meeting, daily meetings, roles allocation, directions of the work process given by the project owner and more. This set of features makes Scrum a valuable platform for projects that require more planning and contracting as the project continues. Training and experience are vital attributes at this stage. The mechanism of planning and reviewing the work which takes place in each sprint is

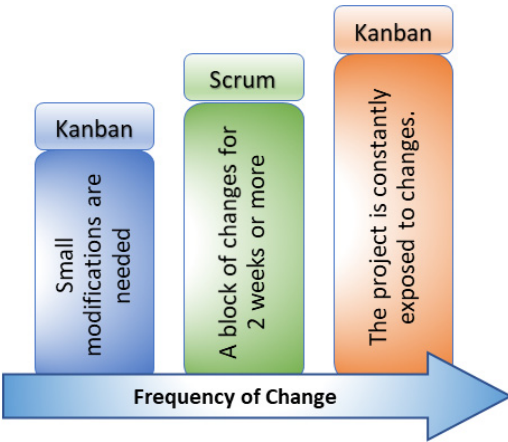


Fig. 8. Preferred system according to the frequency of change.

meant for developing a high-quality management system. This system will also develop the habit of organization and planning among involved team members. Scrum can also give more detailed tracking of progress with the aid of many tools. Sprint report, Burndown chart, epic report, version report, and velocity chart are tools used in story points and planning the next sprints.

In Kanban, a level of communication between different teams or members can be established through signals; however, that cannot be enough in most cases, which makes it more appropriate for the usage of simple independent tasks that do not include complex interactions. It is more of a streamlined for which limited planning is required. As for prioritization, usually, the rule of first in first out (FIFO) is applied. The most one can do with the board is to divide it horizontally into three different levels of importance, or to use cards with different colors indicating a priority system. As for measuring the progress of work, it can be done by monitoring the progress of cards on the board. Figure 9 shows that more tendency toward Scrum is required for more intrinsic tasks.

6.1.4. Level of team maturity

When a team is not mature enough to estimate the time needed for a project to finish or to organize and plan a sprint ahead, it will be unreasonable to set the required features for one sprint since the work might evolve in unexperienced ways. Cases like this might make Scrum less effective and require some organization and experience for team members. Kanban is the right tool for immature teams since it does not require much planning. In Kanban applications, teams can learn more about themselves and their tasks. Kanban can set the ground for a new level of efficiency for an effective Scrum application. After a period (usually 1–1.5 years), teams should be mature enough to understand their weaknesses and should have agreed upon general guidelines for working with different kinds of projects. By then, planning meetings and reviewing the work will be shorter, daily meeting will be more than enough, team members would understand their duties before the project owners give any directions, and less maintenance would be needed because of the sufficient time estimated for features to be developed. These all are indicators of matureness in Scrum team members. At this stage, most of the organizations tend to lean toward being more Kanban again. Figure 10 presents the relationship between the maturity of the team and the correspondence methodology.



Fig. 9. Preferred methodology according to the complexity of the project.

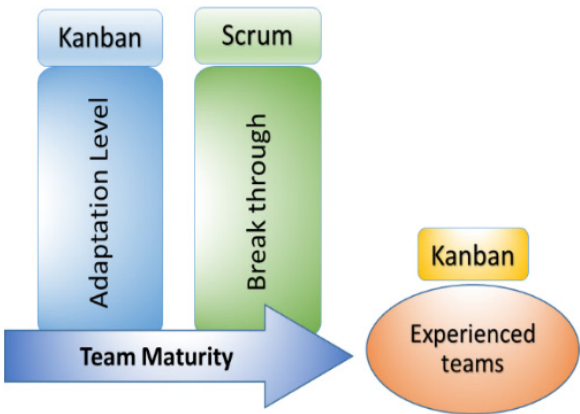


Fig. 10. Preferred system based on the maturity of team members.

6.1.5. *Level of urgency*

The simplicity of Kanban makes it preferable to deal with projects that need to start as early as possible since it does not require any setups. Project managers are required to understand the general principles of Kanban in order to manage the movement of cards on to the board. Scrum requires much more budget in terms of money and time. It also requires training and a lot of adjustments in order to perform.

6.1.6. *Level of uncertainty in the future*

In some cases, the future of the project is not clear in the long run and might be impacted by rapidly fluctuating market dynamics or other factors. Accordingly, many adjustments are expected to take place along the project life cycle. Therefore, more discussion, planning, review, and measurement are needed. Scrum is the ideal methodology in this case. Kanban is not adequate for such situations due to its weakness in operating under complex situations, although it can be helpful to use it with another methodology to focus on the processes flow.

6.1.7. *Size of project*

While in a normal enterprise, Scrum development team is around 8–10 people; larger projects usually require more people involved in the development process. For such a case, Scrum is modified allowing more team members to be involved. This can be done by two methods: the first method is performed by constructing a leading Scrum team which consists of a product owner, a Scrum master, and a development team. Each member in the development team is an operations manager for a subteam (see Fig. 11). The second method is referred to as Scrum of Scrums. In this way, a leading Scrum team which involves the project owner, a chief scrum master, and other team members are constructed. Each team member of the leading team is a scrum master for a subteam which applies Scrum as well. Coordination issues among

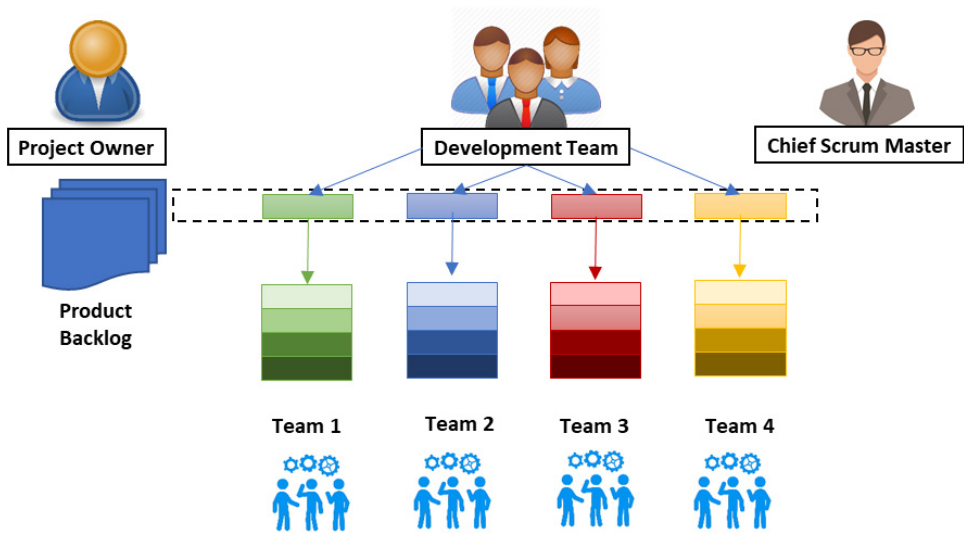


Fig. 11. Scrum modification for several teams.

teams are common in this structure. Also, it is hard for Scrum masters to keep Scrum meetings efficient at this stage. Kanban does not require any project or team size limitations and can perform well in small size as well as large size projects.

6.2. Outcomes of implementation

After implementing Scrum or Kanban, many observations can be detected. Some of these observations are related to the implementation effects on team members and workplace in general. Other observations are related to the project outputs such as productivity and quality.

6.2.1. Effects on the workplace

Scrum can be somewhat rigid and forces people to get more discipline in working with sprint system. For that, Scrum can develop habits of planning, representing the work, communicating with other members, and interacting to form a new concept of sharing information and experiences between people of different backgrounds. Consequently, when an organization decides to start applying Scrum, it should be prepared for some hard period of turbulence and chaos which will continue until the team is familiar with the way things are being done. For Scrum to be successful, it is strongly encouraged to be applied with somewhat organized teams for the level of change it requires.

As for Kanban, it tries to build up some basic concepts, with the idea of continuously improving the system. This improvement is usually with small steps that can be determined when teams understand what works best for the flow of work. It constructs a fixed structure with a systematic point of view for series of tasks to be worked on.

6.2.2. *Effects on the project outputs*

In order to compare measures such as quality, productivity, and risk, both Kanban and Scrum should be applied in the right environments where they can work well. For that, it was most appropriate to check these outputs according to projects that shifted from one of those methodologies to the other and the results were as follows:

- **Quality:** For most of the cases, it was most obvious that people tend to feel more urged to finish their work before the sprint is ended; this usually affected their performance and led to future maintenance to tasks performed at this stage. Although sprint review meetings were a very convenient tool in improving the performance quality of the tasks. Kanban on the other hand does not require such haste and, therefore, resulted in better quality. Also, review meetings of Scrum can have a great replacement in Kanban systems if needed. This replacement is based on the mechanism of continuous improvement or Kaizen which introduces a great tool to improve the overall quality of project outputs. But again, here every case is different, and the output depends mostly on the people involved and their habits in working under pressure.
- **Productivity:** Scrum tends to be more challenging over time; it aims for a set of objectives every time which can create an atmosphere of excitement and motivation for team members. This encourages them to do their best in the process. However, some aspects affect the productivity in various levels. For instance, the time taken by meetings, which can be spent improperly if the time estimation was not as important as getting the work done. With Kanban, there is great attention toward optimizing the line of production. Kanban shines in this area since it highlights the places of bottlenecks and overstocks with an objective to always maintain a small WIP. Since Kanban focuses on the flow of operations, it usually can be more productive when used in the right environment.
- **Accuracy of Delivery:** Teams working according to Scrum can be poor estimators of time in the beginning. After a while, they learn to be better planners and have a good estimation of delivery times. As for Kanban, the time of delivery is usually set based on previous observations of lead times. This can of course give more reliable values, although not always possible.

To sum up, for the provision of theoretical and practical implications, Kanban is easy to understand and implement and simple to launch in practice. Particularly, it is a practical stress-free entry point for non-Agile organizations. Kanban can transform teamwork and demonstrate what Agile can offer. Practically, Kanban is advised for getting started although getting the best out of Kanban is an acquired skill. Kanban can also be the destination or a stepping stone for greater ambitions. Kanban itself is known as a solution for stepping toward Scrum or Agile framework.

On the other hand, in theoretical aspects, Scrum is a framework for addressing complex adaptive problems, while productively and creatively delivering products of the highest possible value. Scrum combines Lean and Agile principles to help teams to deliver products. Scrum is not a project management tool; it is a framework for delivery. Scrum is the most established method of Agile product delivery in practice.






































































7. Conclusion

The innovation of Agile has influenced today's software development market due to its emphasis on continuously reacting to changes and maintaining quality. Agile strategies build the foundation for high-performance teams and consequently thriving business. Studies on Agile methodologies provide a better vision for both users and researchers to understand the key factors of successful applications. This paper was meant to define the similarities and differences of two of the most implemented methodologies within Agile software development, Scrum and Kanban, as well as specifying the suitable environment of implementing each one. A focus group was asked about their ideas according to their personal experiences in both systems. Based on their responses, a set of guidelines were extracted for a better understanding of the strengths and weaknesses of each methodology. Table 1 gives a summary of the collected results.

In conclusion, Scrum is more effective when fundamental change is necessary or when the project is subjected to a specific number of user stories. This methodology puts a high value on organization attributes such as time management, planning, presentation and tracking of work. In Scrum, the team structure is flexible and can be modified according to the vision of the project owner. Kanban is more about the flow of work. This methodology focuses on constructing a streamlined structure; Kanban can adapt to deal with instance changes or different priorities. Kanban is also simple to perform and can be quite efficient when supplemented by a set of quality improvement tools. Furthermore, Kanban can be very practical on already working processes with the need to improve over time, or in continuously manufacturing frameworks.

Also, to some extent, both methodologies are flexible and can be shaped according to work environments where, at some places, a combination of the two

Table 1. Summary of the results.

	Scrum	Kanban
<i>Workplace environment</i>		
Continuous-flow environments		  
Project-oriented environments	  	 
Very limited modifications will be required		  
Blocks of changes will be required	  	 
The project is constantly exposed to changes		  
Projects with complex details and interactions are needed frequently	  	
Unexperienced teams		  
Medium-experience teams	  	 
Experience teams		  
Setup costs (time and training)		  
Large size projects	 	  
<i>Outcomes of implementation</i>		
Engagement of team members	  	
Quality	  	  
Productivity	  	  
On-time delivery	 	  

Note: : avoided; : applicable; : effective.

methodologies which is referred to as Scrumban can be set upon the similarities between Scrum and Kanban. Such a combination is usually set after applying Scrum for some time and the transaction can be sometimes challenging but usually with the proper commitment and organization of team members and by establishing a mechanism for continuous process improvement Scrumban can be quite successful [Nikitina *et al.* (2012)]. Following a continuous process improvement mechanism can be established throughout many different manners. Perhaps, the most popular is the one referred to as Leagile where Agile software development can inherit lean principals to get a whole new different level of performance. Leagile can be reached in six different processes that are well illustrated by Wang *et al.* [2012]. Those different processes differ in the principals of lean they focus on and in their application accordingly.

For further study, it is recommended that more focus should be set on the process of transaction from Kanban to Scrum to Scrumban since this will trace the time and steps of software organizations improvement based on the needed level of performance. This transaction can be tested best using the right metrics for Kanban, Scrum, and Scrumban.

References

- Ahmad, M. O., Dennehy, D., Conboy, K. and Oivo, M. (2018). Kanban in software engineering: A systematic mapping study. *Journal of Systems and Software*, **137**, 96–113, <https://doi.org/10.1016/j.jss.2017.11.045>.
- Ahmad, M. O., Markkula, J. and Oivo, M. (2013). Kanban in software development: A systematic literature review. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, Santander, IEEE, 2013, pp. 9–16.
- Anderson, D. J. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, ISBN: 9780984521401.
- Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001) Manifesto for Agile software development. Available at <https://agilemanifesto.org/> (accessed on 24 January 2020).
- Birgün, S. and Çerkezoğlu, B. T. (2019). A systematic approach for improving the software management process. *International Journal of Innovation and Technology Management*, **16**, 4: 1–28.
- Bjørni, M. A. and Haugen, S. (2019). Challenges with Agile in a system development department: A case study. Master's Thesis, Universitetet i Agder.
- Black, R. and Coleman, G. (2017). *Agile Testing Foundations: An ISTQB Foundation Level Agile Tester Guide*. BCS Learning and Development Limited, The Chartered Institute for IT, Swindon, ISBN: 9781780173368.
- Cockburn, A. (2006). *Agile Software Development: The Cooperative Game*. Addison-Wesley Professional, Boston, USA, ISBN: 0321482751.
- Conboy, K. (2009). Agility from first principles: Reconstructing the concept of agility in information systems development. *Information Systems Research*, **20**: 329–354, <https://doi.org/10.1287/isre.1090.0236>.
- Davis, G. B. (1982). Strategies for information requirements determination. *IBM Systems Journal*, **21**, 1: 4–30, 10.1147/sj.211.0004.

- Dixit, R. and Bhushan, B. (2019). Scrum: An Agile software development process and metrics. *Journal on Today's Ideas - Tomorrow's Technologies*, **7**, 1: 73–87.
- Emelyanova, T., Ilchenko, E., Varlamova, Z. and Paklina, L. (2020). Teamwork management in the field of software development. *Ecological-Socio-Economic Systems: Models of Competition and Cooperation (ESES 2019)*. Atlantis Press, pp. 490–495.
- Highsmith, J. and Cockburn, A. (2001). Agile software development: The business of innovation. *Computer*, **34**, 9: 120–127.
- Hossain, E., Babar, M. A. and Paik, H. (2009). Using Scrum in global software development: A systematic literature review. In *Fourth IEEE International Conference on Global Software Engineering*, Limerick, 2009, pp. 175–184, doi: 10.1109/ICGSE.2009.25.
- Ikonen, M., Kettunen, P., Ozam, N. and Abrahamsson, P. (2010). Exploring the sources of waste in Kanban software development projects, In *36th EUROMICRO Conference on Software Engineering and Advanced Applications*, Lille, 2010, pp. 376–381, doi: 10.1109/SEAA.2010.40.
- Kniberg, H. and Skarin, M. (2010). *Kanban and Scrum-Making the Most of Both*. C4Media Inc., InfoQ.
- Koch, A. S. (2005). *Agile Software Development: Evaluating the Methods for Your Organization*. Artech House Inc, London.
- Kumar, R., Maheshwary, P. and Malche, T. (2019). Inside Agile family: Software development methodologies. *International Journal of Computer Sciences and Engineering*, **7**, 6: 650–660.
- Lei, H., Ganjezadeh, F., Jayachandran, P. K. and Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, **43**, 59–67, 10.1016/j.rcim.2015.12.001.
- Mircea, E. (2019). Project management using Agile frameworks. *Economy Informatics*, **19**, 1: 34–44.
- Martin, R. C. and Martin, M. (2006). *Agile Principles, Patterns, and Practices in C#*. Pearson Education, Inc., ISBN: 0-13-185725-8.
- Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005). Challenges of migrating to Agile methodologies. *Communications of the ACM*, **48**, 5: 73–78.
- Nikitina, N., Kajko-Mattsson, M. and Strale, M. (2012). From Scrum to Scrumban: A case study of a process transition. In *2012 International Conference on Software and System Process (ICSSP)*, Zurich, 2012, pp. 140–149, 10.1109/ICSSP.2012.6225959.
- Ohno, T. (1982). The origin of Toyota production system and Kanban system. In *Proceedings of the International Conference on Productivity and Quality Improvement*, Tokyo, 1982, pp. 3–8.
- Ohno, T. (1988). *Toyota Production System: Beyond Large-Scale Production*. Productivity Press, Portland, OR, p. 29.
- One, V. (2016). 10th annual state of Agile development survey.
- One, V. (2017). The 11th Annual State of Agile Survey.
- Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings of IEEE WESCON*, IEEE, Los Angeles, 1970, pp. 328–388.
- Schwaber, K. (1995). SCRUM development process. In *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications Workshop on Business Object Design and Implementation*, Austin, Texas, USA, pp. 117–134.
- Schwaber, K. and Beedle, M. (2001). *Agile Software Development with Scrum* (Series in Agile Software Development), 1st edn. Pearson Education, Upper Saddle River, NJ, USA, ISBN-10: 9780130676344.
- Schwaber, K. and Sutherland, J. (2017). The Scrum Guide™. Available at <https://www.scrumguides.org/docs/Scrumguide/v2017/2017-Scrum-Guide-US.pdf#zoom=100> (accessed on 5 February 2019).
- Singh, R. (1996). International standard ISO/IEC 12207 software life cycle processes. *Software Process: Improvement and Practice*, **2**, 1: 35–50.

- Smith, P. G. (2005). Balancing agility and discipline: A guide for the perplexed. *Journal of Product Innovation Management*, **22**, 2: 216–218.
- Stoica, M., Ghilic-mica, B., Mircea, M and Uscatu, C. (2016). Analyzing Agile development from waterfall style to Scrumban. *Informatica Economică*, **20**, 4: 5–14, 10.12948/issn14531305/20.4.2016.02.
- Tavares, B. G., da Silva, C. E. S. and de Souza, A. D. (2019). Risk management analysis in Scrum software projects. *International Transactions in Operational Research*, **26**, 5: 1884–1905.
- Tura, N., Hannola, L. and Pynnönen, M. (2017). Agile methods for boosting the commercialization process of new technology. *International Journal of Innovation and Technology Management*, **14**, 3: 1750013.
- Wang, X., Conboy, K. and Cawley, O. (2012). “Leagile” software development: An experience report analysis of the application of lean approaches in Agile software development. *Journal of Systems and Software*, **85**, 1287–1299.
- Winter, H. (2019). Managing requirements. *The Business Analysis Handbook: Techniques and Questions to Deliver Better Business Outcomes*. Kogan Page Limited, London, pp. 179–183.
- Womack, J. P., Daniel, T. J. and Daniel, R. (1990). *The Machine That Changed the World: The Story of Lean Production*. Harper Collins, New York, NY.
- Yordanova, S. and Toshkov, K. (2019). An Agile methodology for managing business processes in an IT company. *Business Management*, **3**, 27–90.

Biography

Wael Zayat is a PhD student in Industrial Engineering in Marmara University, Turkey. He is studying the applications of NP-hard combinatorial optimization using metaheuristics. Concurrently, he is a software developer who works with database management systems in a tourism agency in Istanbul, Turkey.

Ozlem Senvar (PhD in Industrial Engineering) is an Associate Professor at Marmara University in Department of Industrial Engineering. Ozlem Senvar (PhD) was officially granted (Ref.2013/327-350) by European Commission for her post doctoral researches in France. She worked in Logistics and Optimization of industrial Systems of Institute Charles Delaunay at Universite Technologie Troyes. She has been author of many peer reviewed international articles, book chapters, and conference papers. Her research interests involve Production and Operations Management, Advanced Quality Engineering, Statistics, Machine Learning, Artificial Intelligence, Data Mining, and Decision Making.