Kanban:
Visual Cards for Software Engineering

By David Wilson

Introduction

According to the website for the Agile-engineering consultancy Kanbanize, the word “Kanban” is Japanese for “visual card.” [1] This is not only an appropriate translation but an informative one; it immediately brings to mind both the signature tool in its kit, the Kanban card, and the most important aspect of its management approach, the visualizing of the engineering workflow to understand the weight of each assigned task. Kanban grew out of a lean manufacturing strategy developed by the Toyota company shortly after World War II. In 2010, a Microsoft software engineering manager named David J. Anderson released a book recognized as the key to bringing the Kanban system to the software development world. [2] A 2015 article in *Robotics and Computer-Integrated Manufacturing* found that 40% of respondents to an online survey had integrated Kanban into their software development systems. [3] But what is Kanban? How does it work, and where does it fit into the Agile philosophy?

This paper will explore the Kanban concept both in general terms and in specific as it applies to software development. It will explore the history and methodology of Kanban; where it fits into the Agile-engineering spectrum, including how it stacks up against and ultimately supports the popular Scrum development model; a real-world success story with Kanban; and the drawbacks of the Kanban system. The paper will conclude with the author’s final thoughts on the system.

History of Kanban

Kanban developed from the Toyota Production System (TPS), a strategy for lean manufacturing developed by the Toyota company in the late 1940s. [2] TPS, depicted in Figure 1, made use of a prototype of the iconic Kanban cards. These cards contained task descriptions for a given project under management. They were hung on racks, and the position of each card on the rack communicated information about the status of the associated task as well as the number of tasks at the same position within the manufacturing workflow. The purpose of this system was to help establish the “Just-in-time” philosophy of manufacturing, an idea that emphasizes the reduction of waste in the manufacturing process by focusing attention on the requirements of each task and how those requirements can be met without exceeding them. Another key focus of TPS, one that would become key to Kanban as well, was the concept of continuous improvement. [4] The so-called “Toyota Way” emphasizes the need to observe and adapt their processes to forever aim for less wasted effort and fewer wasted resources.

Figure 1: The original Kanban, the Toyota Production System. [5]

The 2010 book *Kanban, Successful Evolutionary Change for Your Technology Business*, by David J. Anderson, adapted the Kanban concept from manufacturing to software development. In his book, Anderson states that his two main objectives in this effort were to (1) encourage the pursuit of a sustainable pace for software engineering, fighting crunch time and burnout among developers; and (2) improve change management capabilities for companies with a multitude of development teams working on different projects. [6] On page 25 of the Kindle Edition of this book, Anderson writes that implementing a prototype version of the Kanban system at Microsoft caused dramatic effects: “With very little resistance, productivity more than tripled and lead times shrank by 90 percent, while predictability improved by 98 percent.”

Methodology of Kanban

Anderson identifies Kanban as a kind of “pull system” approach to process management. He defines this in the second chapter of his book as meaning that “new work is pulled into the system when there is capacity to handle it, rather than being pushed into the system based on demand." [6] Kanban represents this work as a series of cards that each depict individual tasks. These cards are moved between the columns on a Kanban board, which each represent a different stage of the workflow, as shown in Figure 2. One can glance at the board to see how many tasks are engaged in each phase of the process. Setting a maximum number of cards per column or per board creates a limit to how many tasks the team can be assigned at once, providing a simple guideline that both informs management decision-making and insulates developers against unsustainable workloads.

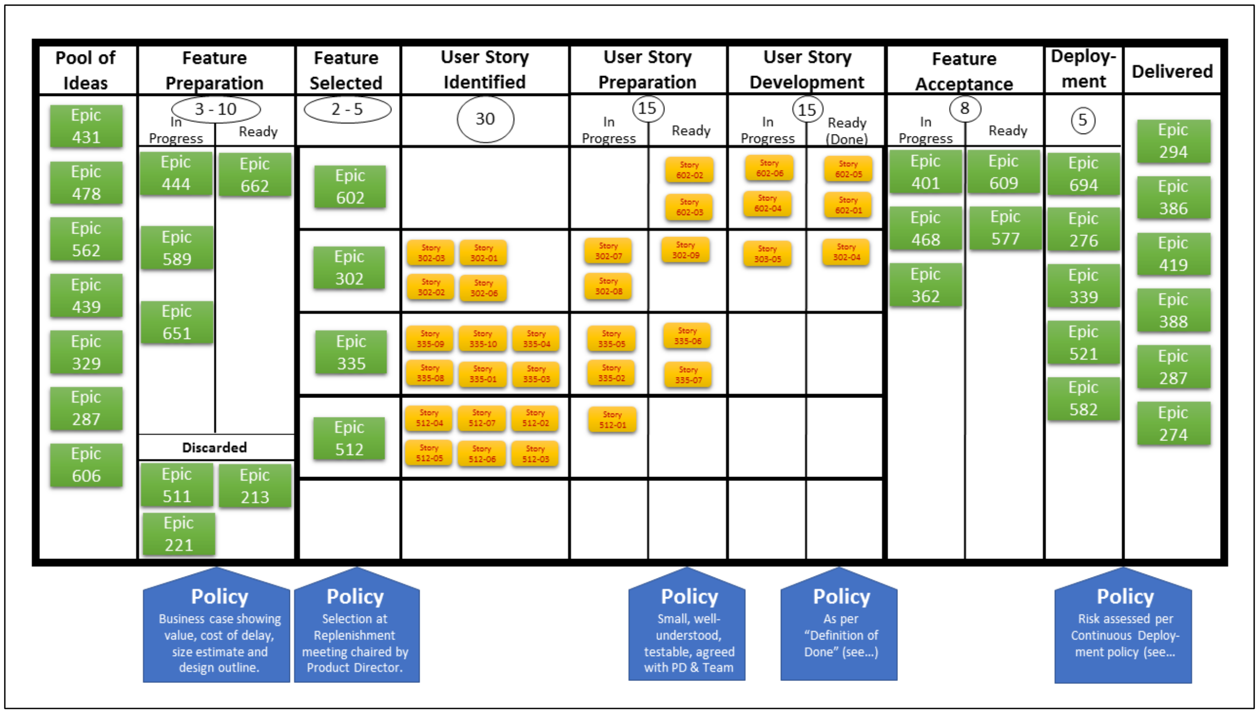


Figure 2: Example Kanban Board with cards. [2]

Watching these task cards move through the workflow also provides a powerful visual aid that allows the manager to better understand not only the status of the individual tasks but the weaknesses of the process itself. If one column seems to accumulate cards faster than others, this is an indication that the associated phase of the workflow might be a bottleneck to the overall process. This allows the process manager to pinpoint at a glance the areas of the workflow most in need of improvement.

Kanbanize recognizes the following four principles as the core of Kanban philosophy:

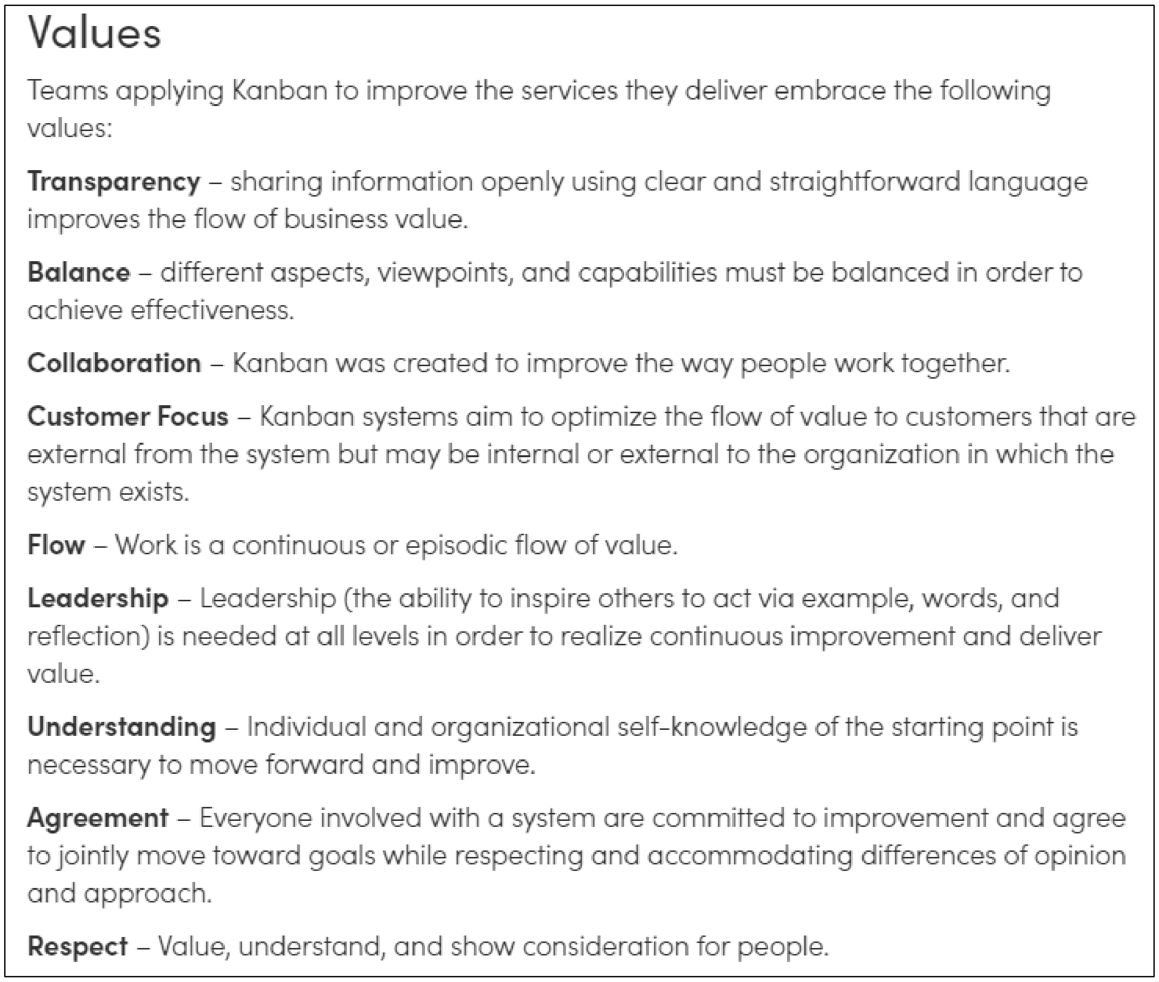
1. "Start with What You Do Now," which means that Kanban is designed to be overlaid on existing processes so that they can be analyzed without disrupting them.

2. “Agree to Pursue Incremental, Evolutionary Change,” which means that Kanban is designed to meet minimal resistance and encourage small, continuous improvements to the process.
3. “Respect the Current Process, Roles, and Responsibilities,” which means that Kanban recognizes the value in the organization’s current workflow and personnel and seeks to improve them rather than trigger fears of outright substitution.
4. “Encourage Acts of Leadership at All Levels,” which means that Kanban promotes the idea that leadership comes from everyday acts on the front line of each team and that fostering a mindset of continuous improvement in all personnel leads to optimal performance. [5]

Overall, these principles combine with the “physical” tools of Kanban, i.e., the cards and the board, to place the focus on continuously improving an existing software development workflow through visual understanding, communication between team members and management, and sustainable workloads. However, as Figure 2 shows, the Kanban system no longer requires tangible cards and boards. It can instead be implemented through digital tools that can be shared with an entire dev team simultaneously without requiring anybody to leave their computer. Many such tools are now available for free on the Internet, permitting even independent developers without a budget to manage their processes with Kanban. This includes such tools as Atlassian’s Trello [7] and Kaleidos’s Taiga. [8]

Kanban on the Agile-Engineering Spectrum

Kanban’s focus on sustainable pacing for developers, simple decision-making for managers, and continual process improvement place it neatly into the spectrum of Agile methodologies. The Agile Alliance lists the following values it associates with the Kanban system: transparency, balance, collaboration, customer focus, flow, leadership, understanding, agreement, and respect. [9] These features are further defined in Figure 3, a screenshot from the Agile Alliance’s webpage on Kanban.


Values

Teams applying Kanban to improve the services they deliver embrace the following values:

Transparency – sharing information openly using clear and straightforward language improves the flow of business value.

Balance – different aspects, viewpoints, and capabilities must be balanced in order to achieve effectiveness.

Collaboration – Kanban was created to improve the way people work together.

Customer Focus – Kanban systems aim to optimize the flow of value to customers that are external from the system but may be internal or external to the organization in which the system exists.

Flow – Work is a continuous or episodic flow of value.

Leadership – Leadership (the ability to inspire others to act via example, words, and reflection) is needed at all levels in order to realize continuous improvement and deliver value.

Understanding – Individual and organizational self-knowledge of the starting point is necessary to move forward and improve.

Agreement – Everyone involved with a system are committed to improvement and agree to jointly move toward goals while respecting and accommodating differences of opinion and approach.

Respect – Value, understand, and show consideration for people.

Figure 3: The values of the Kanban system, per the Agile Alliance. [5]

On the same site, the Agile Alliance discusses the 12 Principles of the Agile Manifesto, which includes ideas such as (1) customer satisfaction through early and continuous delivery of software, (2) give motivated individuals the environment and support they need and trust them to get the job done, and (3) promote sustainable development. [10] Kanban's function as a visual aid to identify process bottlenecks helps the team keep the deliverables flowing, while its inclination to meter additional work to that "pulled" by capability rather than pushed by demand helps maintain a sustainable pace of development. These factors converge to promote a comfortable working environment for employees. In this way, Kanban seems like a natural fit for teams that follow the Agile Manifesto.

What's more, Kanban's principle of "Start with What You Know" emphasizes the compatibility of Kanban with existing processes. It is intended to be implemented over the top of a team's current workflow to improve their ability to analyze and improve that workflow. As a result, Kanban can be treated as a complement, rather than a replacement, to existing software development models.

Scrum is one of the more popular Agile methodologies, identified primarily by the continuous cycle of "sprints," reviews, and product iterations, as demonstrated in Figure 4. In short, a contact designated as the "Product Owner" manages communication between the development team and the customers and converts the customer's expectations into a document called the "Product Backlog." The team then takes a selection of work from that backlog and focuses down on that work over the course of a limited-duration development iteration called a "sprint." These iterations are conducted alongside short, daily meetings with a designated "Scrum Master" to keep the team on track and to adapt the process to overcome hurdles. [11]

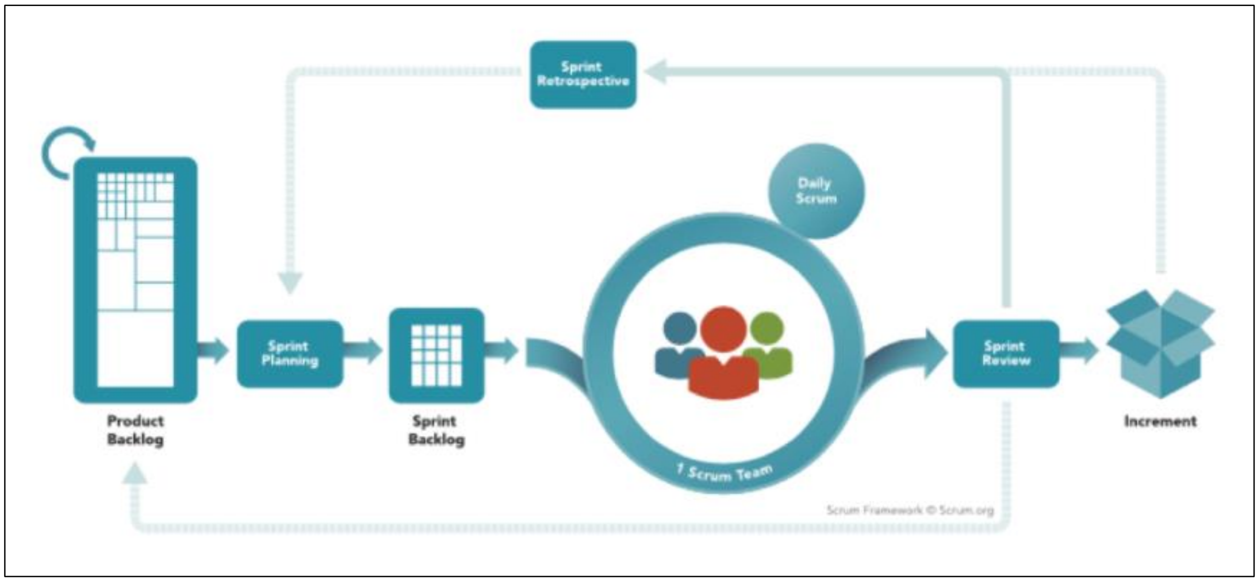


Figure 4: The Scrum Framework, courtesy of Scrum.org. [11]

Kanban and Scrum share an emphasis on continuous process improvement, controlled workloads, and team communication. The two systems differ on two key points, specifically: (1) Kanban emphasizes continuous flow and delivery, while Scrum breaks flow into defined sprints and places delivery at the end of each sprint; and (2) Kanban permits process adjustment at any given time, while Scrum typically avoids process changes during sprints to prevent any possible reduction in productivity. [12] However, if these differences can be reconciled, the two methodologies synchronize well.

The two systems can also coexist on a quantitative level. Scrum makes use of a metric called **velocity**, which is defined as the "average amount of Product Backlog turned into an increment of product during a sprint by a Scrum Team." [13] In other words, velocity measures how many tasks a team can deliver in a given period of time. Kanban uses a metric called **Lead and Cycle time**, which measures the average time it takes to complete a given task. [2] These two metrics give the manager a view on the same general idea from two opposite directions, and their convergence presents a powerful tool: a standard for the amount of work you can expect to accomplish given a unit of time and a standard for the amount of time you can expect a given unit of work to take. This information not only gives the manager insight into the team's efficiency and upper workload limit, but it also provides a simple and reliable way to measure and eventually predict how fluctuations in policy and personnel will affect the team's ability to deliver quality software on time and as expected.

At least in theory, Kanban and Scrum united can form a stronger system for an Agile-focused software team than either methodology on its own. But how does Kanban work in action? How have companies in the real world benefited from incorporating Kanban into their existing workflows?

Success with Kanban

Canadian software news website InfoQ published a 2014 article exploring the success of the Kanban system in action at Siemens Health Services. [14] Six years after adopting the Scrum methodology, Siemens found that their software development teams struggled to meet deadlines. The article states, "The last week of each sprint was always a mad rush by teams to claim as many points as possible, resulting in hasty and over-burdened testing." Essentially, the development team had to crunch to finish their objectives over each sprint, and this resulted in sloppy software that got stuck in extended testing periods to compensate for all the defects. However, because these tasks were still technically getting done, the Scrum standard metric of velocity still presented a rosy image. This, in turn, meant that nothing changed between sprints to make these tasks more manageable.

It wasn't until December of 2011 that management at Siemens elected to add a layer of Kanban to their Scrum system. They put a "Flow Team" into place to design and implement Kanban among the developers. The eventual result was a drastic reduction in both the average number of defects per sprint and the gap between the average numbers of defects detected and the number resolved. This is demonstrated in Figure 5, where the colorful lines each represent an average number of defects detected or resolved before or after Kanban implementation as follows: (1) purple shows defects detected before Kanban, (2) red shows defects resolved before Kanban, (3) green shows defects detected after Kanban, and (4) blue shows defects resolved after Kanban. [14]

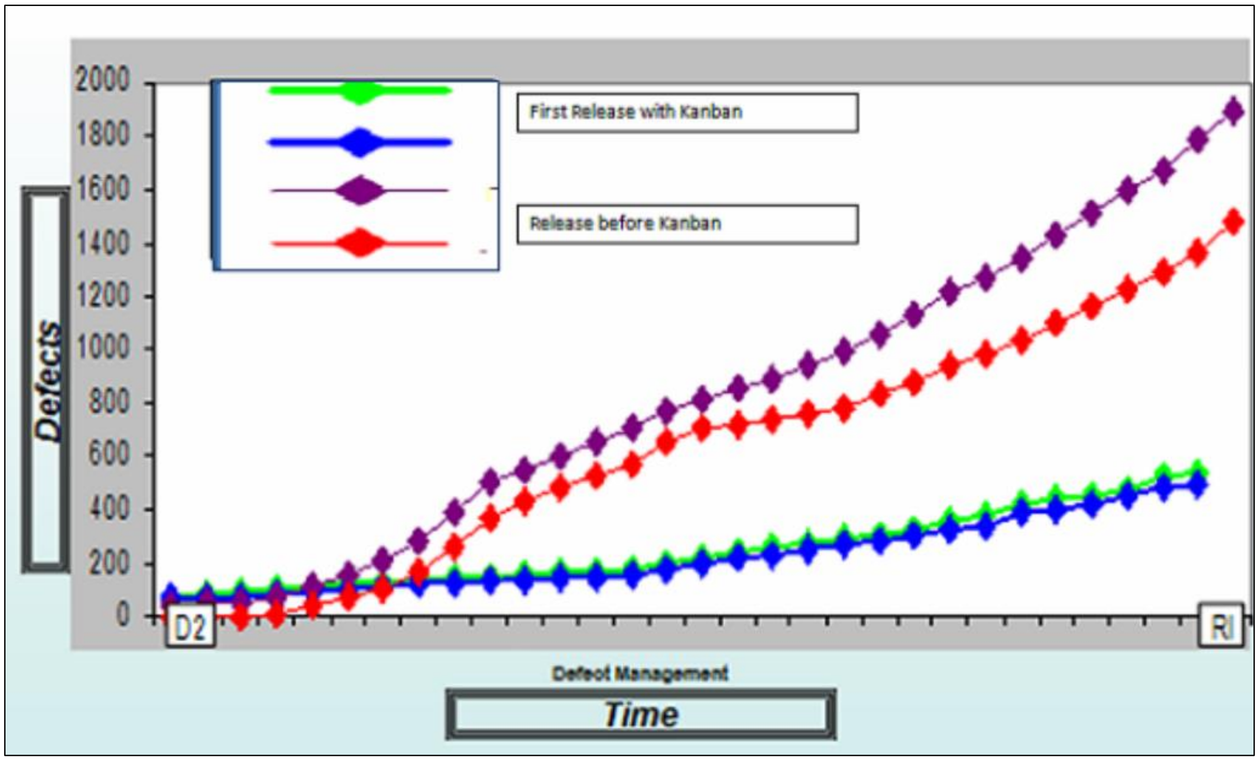


Figure 5: A representation of (purple) defects discovered before Kanban, (red) defects resolved before Kanban, (green) defects discovered after Kanban, and (blue) defects resolved after Kanban, per release at Siemens. [14]

The drop in defect count combined with the improvement in the ratio of defects discovered to defects resolved make it clear that the combination of Kanban and Scrum was a powerful solution to the problems Siemens was facing. This shows that the addition of the Kanban methodology can benefit even a company that already employs an Agile software development model.

Drawbacks of Kanban

Kanban is not without its issues. Due to its simplicity, these problems typically arise from its implementation rather than the system itself. One big threat is the failure to respect the limit on the number of work-in-progress tasks that can be placed on the board at once. [15] As a pull methodology, one of Kanban's primary focuses is on maintaining a sustainable pace for the development team by limiting the amount of work they must shoulder at once. Ignoring this limit defeats one of the most important aspects of the Kanban system, and this can easily make the system fail.

Another potential issue for Kanban is the desire on management's part to see immediate results. [15] Kanban emphasizes small, incremental but continuous improvements to the process. This means that companies won't see results like in the Siemens case overnight or even within the first year. With

Kanban, the key is monitoring the process and adjusting it gradually to reduce resistance and discomfort. This also means that Kanban may not be the best fit for a company that cannot wait for its software processes to improve with time. Kanban is like a honing stone in this way; it can sharpen development processes that are resilient enough to undergo the treatment, but it cannot fix a process that is too damaged to survive. In these cases, a more drastic overhaul might be necessary. However, once a new software process is in place, Kanban may yet complement that process well.

Final Word

To summarize, Kanban is a system of simultaneous project and process management. It allows the development team to control a sustainable flow of tasks through the system and the management team to identify and overcome bottlenecks in the software process. Kanban is commonly considered a member of the Agile school of development [12], but it can complement other Agile methods to great effect. It has its drawbacks, but these issues relate more to its implementation than to the system itself.

Overall, Kanban is a fantastic tool for managing the software development process. With many free examples of Kanban software available for public use [7] [8], and with impressive results for large companies that have implemented Kanban to great success [14], a strong argument exists for at least testing this methodology at every level of software development from the independent game designer to the corporate juggernaut. Kanban makes an excellent addition to any software development toolkit.

References

[1] Kanbanize, "What is a Kanban Card? All You Need to Know," Kanbanize, 2021. [Online]. Available: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban-card. [Accessed 04 March 2021].

[2] Wikipedia, "Kanban (development)," Wikimedia Foundation, Inc., 2021. [Online]. Available: https://en.wikipedia.org/wiki/Kanban_(development). [Accessed 04 March 2021].

[3] H. Lei, F. Ganjeizadeh, P. K. Jayachandran and P. Ozcan, "A statistical analysis of the effects of Scrum and Kanban on software development projects," *Robotics and Computer-Integrated Manufacturing,* no. 43, pp. 59-67, 2017.

[4] Wikipedia, "Toyota Production System," Wikimedia Foundation, Inc., 2021. [Online]. Available: https://en.wikipedia.org/wiki/Toyota_Production_System. [Accessed 04 March 2021].

[5] Kanbanize, "What is Kanban? Explained in 10 Minutes," Kanbanize, 2021. [Online]. Available: https://kanbanize.com/kanban-resources/getting-started/what-is-kanban. [Accessed 04 March 2021].

[6] D. J. Anderson, Kanban, Successful Evolutionary Change for Your Technology Business, Blue Hole Press, 2010.

[7] Atlassian, "Trello, Inc.," Atlassian Corporation Plc, 2021. [Online]. Available: https://trello.com/home. [Accessed 04 March 2021].

[8] Taiga Agile, LLC, "Taiga: Your opensource agile project management software," Kaleidos Open Source SL, 2021. [Online]. Available: https://www.taiga.io/. [Accessed 04 March 2021].

[9] Agile Alliance, "What is Kanban?," Agile Alliance, 2021. [Online]. Available: https://www.agilealliance.org/glossary/kanban/. [Accessed 04 March 2021].

[10] Agile Alliance, "12 Principles Behind the Agile Manifesto," Agile Alliance, 2021. [Online]. Available: https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/. [Accessed 04 March 2021].

[11] Scrum.org, "What is Scrum?," Scrum.org, [Online]. Available: https://www.scrum.org/resources/what-is-scrum. [Accessed 04 March 2021].

[12] M. Rehkopf, "Kanban vs Scrum," Atlassian Corporation Plc, [Online]. Available: https://www.atlassian.com/agile/kanban/kanban-vs-scrum. [Accessed 04 March 2021].

[13] P. Doshi, "Agile Metrics: Velocity | Scrum.org," 17 May 2018. [Online]. Available: https://www.scrum.org/resources/blog/agile-metrics-velocity. [Accessed 04 March 2021].

[14] B. Vallet, "Kanban at Scale - A Siemens Success Story," 28 February 2014. [Online]. Available: https://www.infoq.com/articles/kanban-siemens-health-services/. [Accessed 04 March 2021].

[15] Kanbanize, "6 Reasons Why You Failed to Implement Kanban," 2021. [Online]. Available: https://kanbanize.com/blog/problems-with-kanban-implementation/. [Accessed 04 March 2021].