Lean Principles in Software Development Projects

Nela Cvetković
Department of Industrial Engineering and Management
Faculty of Technical Sciences
Novi Sad, Serbia

Slobodan Morača
Department of Industrial Engineering and Management
Faculty of Technical Sciences
Novi Sad, Serbia

Abstract—**With a recent noticeable shift towards Lean Kanban software development and increase of its popularity, certain challenges and need for further research and analyses stand out as well, in order to provide better understanding and support, considering the lack of existing literature and case studies on this topic. With that aim, we have presented our point of view and perspective on significant Kanban principles and practices relevant for software development, certain approach to its implementation and main motivational factors, benefits and challenges having an impact on Kanban introduction to software development.**

Keywords-Lean; Kanban; Agile; Software development; Scrumban;

I. INTRODUCTION

Most of the software products are specifically tailored with intents to fulfill customers' requirements and needs as successfully as possible. However, only few of the projects intending to deliver these products are defined as successful at the end of its implementation; the deliverables of low percentage of software development projects are recognized as satisfactory by the customers or to have generated additional value to the users. According to the research [1], only 29% of all size projects have been defined as successful, 52% as challenged, while 19% have been defined as failed. Although supporting the attitude that great impact on the results are having human resources and the knowledge and experience they posses, this fact points out the necessity for project management in IT projects and its constant improvement and adaption in order to overpass the constraints and internal or external disruptive factors. Alongside of the major project constraints according to PMBOK [2]: scope, budget, schedule, risk, quality and resources, the most significant factors enhancing the complexity of managing IT projects are: continuous changes of underlying technologies, complexity of projects disabling realistic schedule and budget estimations, ineffective project management skills, lack of clients' involvement and clear requirements [3]. Furthermore, one of the greatest preconditions for successful implementation of IT projects is extreme detail of design and frozen design [4]. In order to provide changeless design before the start of implementation, no alterations of requirements and specifications can be accommodated, which is proved to be impossible case in business practice.

Organizations and practitioners have been trying to eliminate these problems and challenges through constant improvement and modification of software development methodologies. However, it is noticed that with every new methodology, despite certain challenges are being overcome, the other obstacles occur, again requesting the new modifications. Thus, as a response to so-called "heavyweight" or traditional methodologies, which were said to entail extensive planning, up-front analyses and design [7], to be robust, bureaucratic, process-centric [8], to assume the team has almost complete and perfect information about project requirements, causing changes not to be encouraged and extremely expensive [9] etc, Agile methods have been introduced to software development. Agile methodologies have proved to be adequate and useful when it comes to the greatly important preconditions for successful projects: ability to respond and react to the changing needs of customers and to reduce delivery time [10]. Furthermore, these methodologies help avoid robust documentation, encourage personal communication, attach legitimate importance to the stakeholders and allow them to affirm their requirements through constant verifications during the project implementation period. Nevertheless, it cannot be claimed that Agile methods are providing all the needed solutions and benefits without expressing few not so strong points. The time-boxed processes [11], exhausting work scheduling, and non applicability on large-scale projects, represent their most important demerits.

Next approach that has drawn attention as suitable for addressing these factors was Lean approach as new wave in software development [10]. The great variety of projects and the increased complexity required the engagement of cross functional teams. Consequently, the exchange of knowledge, methods and tools characteristic for certain professional areas were increased. After proving to be very successful in manufacturing industry, certain principles of Lean approach have been applied to software development for continuous agile process improvement [11]. Software developers have recognized various advantages of Lean approach, among which the most significant are: optimization of process flow, ability to deliver value to the customer more efficiently by finding and eliminating waste [10] and pull system. Lean approach that so far has been introduced in software development is Kanban. Kanban is seen as a continuous, flow-based substitute to time-boxed agile processes [11].

However, despite the increase of interest in Kanban, and generally, Lean approach in software engineering, there is still a lack of literature or real case studies on this topic. The goal of this paper is to present the new view and perspective on Kanban practices and its implementation in software development, based on analyses of existing literature on current

state-of-art regarding the usage of Kanban in software engineering and real life experience regarding Traditional, Agile and Lean approach.

II. LEAN APPROACH IN SOFTWARE DEVELOPMENT

Lean thinking has its roots in Toyota (then called Toyota Production System) where has been created in the mid-70s. Since then, Lean has been recognized as the mainly operational management strategy [12] in manufacturing sector. However, so far its ideas have been also successfully implemented in service and product development [13].

Moreover, recently the agile community has started to look toward Lean software development approaches [11], assaying the usage and benefits of Lean principles known for manufacturing and production industry on software development process. According to [20] there are claims that Lean software development provides the theory behind agile practices [11]. While some see Lean software development as an instance of agile methods [14], others claim that Lean software development is rather a method category in itself [15]. We hold on opinion that, although Agile and Lean share certain foundations and principles, Lean has its own specific features not represented in Agile approach and not supported by it, wherefore we see Lean as next generation of software development approach. From case to case, there are different motives for implementation of Lean approach in software development. Since, on the one hand, many large companies using Lean techniques in their manufacturing operations have found it beneficial in many aspects, and on the other hand, have recognized software used in their company to be the substantial part of their products and business, they are seeking to transfer their successful lean experience to the software developers [13]. Certain organizations using Agile methods (of which the most dominant is still Scrum (56%), according to the last State of Agile Annual Survey [16]) have been encountering a problem when it comes to applying Agile methods at scale. Organizations having trouble connecting the work of large-scale projects using only Agile techniques are identifying probable solutions in Lean principles.

As already said, the main benefits of Lean development are improved efficiency in eliminating waste (including by that elimination of everything that is not creating value) and increased efficiency in delivering value to the customers [10]. Additionally, pull system characteristic for Lean approach has proved to be essential for process flow optimization, where system can act as event-driven, producing and delivering what customer wants and where he wants, instead of producing in advance for inventory. Related to that, as the main overall goal of Lean development can be identified achievement of continuous and smooth flow of production with maximum flexibility and minimum waste in the process [17].

Lean concept involves three main elements: lean concepts, lean principles and lean practices [11]. Lean concept is providing organizations with possibility to "specify value, line-up value-creating actions in the best sequence, conduct these activities without interruption whenever someone requests them, and perform them more and more effectively" [18]. There are five guiding lean concepts:

- Value and clear understanding of what it is;
- Value stream;
- Process flow;
- Pull system and
- Perfection and constant strive for it [18], [19].

Concerning software development, the relevant items that can be defined as waste that are tended to be eliminated are: "extra features, waiting, task switching, extra process, partially done work, movement, defects and unused employee creativity" [11], [20].

Poppendieck & Poppendieck [20] have adjusted lean principles into lean principles relevant to the software development:

- Eliminate waste;
- Build quality in;
- Create knowledge;
- Defer commitment;
- Deliver fast;
- Respect people;
- Optimize the whole.

Although these principles are expressing the core of Lean software development, they are not enough for complete understanding. Therefore, it is necessary to observe these principles, but also other identified elements. Related to that, we present practices specified by authors [11] in their paper, which are considered to be lean practices relevant for software development. The list is offering the starting point for further development. As the lean practices relevant for software development are identified:

- Address bottlenecks;
- Defer decision making;
- Develop rewarding system;
- Workload leveling;
- Kaizen;
- Kano analysis;
- Use pull system;
- Kanban board;
- Limited WIP;
- Value stream mapping, etc.

One of the most popular instances of Lean approach is Kanban for which it is considered to offer a less prescriptive element as compared to Agile methods [15], [21].

III. KANBAN IN SOFTWARE DEVELOPMENT

Kanban was developed by Taiichi Ohno, an industrial engineer at Toyota, as a system to improve and maintain a high level of production. It was one method to achieve Just In Time production [22]. Kanban (kahn-bahn) is a Japanese word literally meaning "visible record" [23]. In manufacturing it refers to Kanban cards where one card is associated with each piece of work. It represents one of the manufacturing strategies and tools in Lean manufacturing systems where it may control the levels of buffer inventories in the system to regulate production. When a buffer reaches its preset maximum level, the upstream production activities are not being continued for that particular part type, that way achieving minimum inventory at any one time [24]. That is the main Kanban feature that practitioners have recognized as significant for software development. Furthermore, one of the most important premises of Kanban is that system requires production only when the demand of products is available. Kanban has proven to be useful and helpful tool for an organization, enabling advantages from improvement of productivity and minimization of waste [24], development of flexible work stations, to minimization of waiting time and logistics costs [23]. In other words, the idea of Kanban approach is "to use visual signals to synchronize the flow of work with process capacity, limit the waste of work interruption, minimize excess inventory or delay due to shortage, prevent unnecessary rework, and provide a means of tracking work progress" [25].

Recently, with the application of Lean approach in software development, previously described Kanban principles have been tried to be "replicated" from manufacturing to software development. It can be said that, even developers have seen the potential of Kanban and its principles based on successful use in manufacturing sector, its success in software development could not be guaranteed. First of all, Kanban systems in manufacturing refer to managing and handling production pieces, "physical" and tangible items, while the software development process includes non tangible items and the main focus is on how to manage human resources and their efforts. Secondly, not only IT organizations imply different needs and specifities, but also the projects inside those organizations demand very diverse approaches, complicating the decision making process upon methodology to be used and raising the chances of not achieving the desirable results. These differences between two industries emphasis the need for researching the Kanban in software development and providing the real case studies in order to identify the most successful way to adjust Kanban and its principles to software development process. The significant fact is that in the last 5-6 years Kanban has being applied to software engineering and is becoming the key Lean practice in this field [10]. According to [16] the Kanban is on the fifth place when it comes to the usage of Agile methodologies (5%).

A. Kanban software development background and motivation for Kanban usage

In software development, the main focus of Kanban is "to accurately state what work needs to be done and when it needs to be done, by prioritizing tasks and defining workflow as well as lead-time to delivery"[26]. That means Kanban represents certain flow control mechanism for pull driven development. In brief, Kanban aims to provide visibility to the software development process, communicate priorities and highlight bottlenecks [27] which results in a constant flow of releasing work items to the customers, as the developers focus only on those few items at a given time [28]. Regarding that, Kanban systems in software development have special significance from the aspect of scheduling work. Related, Kanban systems bring differences compared to Agile methodologies. They are focusing on continuous flow of work and disregard fixed iterations. Kanban, as in the case of Agile methodologies, sets the requirements and related work items and implements them incrementally. However, instead of time-boxed iterations, the Kanban team chooses those work items and starts working on them when that is needed and there is capacity. The team develops features one after the other and as soon as it is ready, the team works only on one or very few at a time [10]. In other words, the Kanban in the context of software development allows to the teams to visualize the workflow, limit WIP at each workflow stage and measure the cycle time [29].

In their study [27] the authors identify main motivational factors for adopting Kanban in software development projects:

- Improvement of team communication;
- Improvement of development flow;
- Reduced lead time;
- Increased productivity;
- Better transparency in organization and of work;
- Better control of flow; focus on flow and absence of fixed iterations.

So far, literature and the results of real life case studies have confirmed that these motivational factors can be achieved on significant level with Kanban. However, despite the enhanced expected benefits and motivational factors, there are challenges in adoption of Kanban approach which should be mentioned. Once staff is used to the current methodologies, it is difficult to motivate them to accept changes. The big impact on that also has organizational culture [27]. Additionally, the lack of specialized skills and training [33], as the experience with Kanban and knowledge on how to adjust it to software development, represent highly significant obstacles in adapting Kanban. What seems as a reasonable solution is providing relevant trainings, engagement of experienced staff and trainers and approaching the implementation of Kanban from the bottom, meaning, having the impact on every individual in organization and making sure they understand what Kanban is and how it contributes to the organization.

B. How to implement Kanban

Once organizational environment is adjusted and dismissive attitude is neutralized as much as possible, Kanban should be set. Despite the increase of Kanban usage and its current popularity, followed by lack of literature and case studies, there is also lack of standardized or widely accepted Kanban implementation process and knowledge on how to transform the approach for manufacturing sector to the IT sector. Each situation had its specifities and usually every

organization demands different solutions for different challenges.

So far, very few authors have tried to approximate and describe possible "scheme" for implementation of Kanban in software development process. What we find worth representing are steps typically included in setting up a Kanban system in practice and relevant for Kanban in software development projects, described by Corona & Pani [10]. Additionally, Ladas Corey [30] in his book, besides describing Kanban as a possible convenient match for Scrum method, supports the following steps for implementation of Kanban system by Kanban axioms stated by him. Those steps include:

- Mapping the flow and identifying the activities in it;
- Defining the requirements and their expression through a set of features;
- Devising a maximum limit for the features under work in each activity while taking into consideration the activities and team composition;
- Setting-up the Kanban board and definition of specific features regarding the board;
- Devising relevant policies (such as dealing with block, achieving WIP limits, etc.);
- Defining the formal way of performing meetings;
- Devising the release management;
- Defining tools and additional methods to be used.

C. *Kanban software development principles and practices*

Observing Lean software development principles [20]: build quality in, create knowledge, defer commitment, deliver fast, respect people and optimize the whole and Kanban software development principles specified by [26] it can be noticed that there is the shared grounding and foundation. Defined basic Kanban software development principles [26] are:

- Limit work in progress;
- Visualize the workflow;
- Measure and manage flow;
- Make process policies explicit;
- Improve collaboratively.

Limiting work in progress: This principle may be specified as one of the most valuable principles of Kanban approach in software development and it is closely connected to the idea of pull systems. It focuses on limiting work in progress according to capacity. This principle secures that work cannot be started until there is an available appropriate resource, characterizing the mechanism as a pull system, "since the work is pulled into the process rather than pushed via a schedule" [25]. Limiting WIP has its significant influence in flow control and enhancement of value by limiting the amount of work that is being assigned to the resources. Correctly defined WIP limit ensures there will not be overloads and that sustainable pace of development will take place. It quickly brings to light issues that impair performance: "when work cannot move forward because the WIP limit has been reached in the next state, it makes the current constraint on the system highly visible, thus forcing the team not to take more work until the problem with the constraint is fixed" [31]. The goals limiting WIP is accomplishing are: acceleration of value by completing higher value before starting lower value work [25] reduced lead time [31], decreased number of analyzed tasks which stayed too long in the implementation stage [32] and continuous and "smooth" flow [29]. This small improvement is strongly encouraged, since it can improve the development process by helping avoid bottlenecks in all stages.

Visualization of the workflow: The visual representation of the work is critical to Kanban success, since it provides immediate understanding of the state of flow through the set of process activities [25]. This way the occurred process delays or resource issues are highlighted and visible, enabling the team to react immediately in order to resolve the problem. In order to enable the visualization work items are presented on a Kanban board serving as a visual control mechanism. The workflow through various stages of development process is indicated. Various stages are represented by columns on the board and items are represented with cards. As work progresses through the development lifecycle, the cards move from first column on the left side towards the last column on the right side. Related to the pull system characteristic for Kanban, when a card is completed in one column, it moves to the next, that way creating an open space in its current column and allowing to the team to pull a completed card from a previous column [31]. Due to the existence of columns, i.e. queues, the costs of possible delays or other unpredictable and usually invisible aspects of scheduling become integral inputs to decision making [25].

Measure and manage flow: This principle points out the necessity of measuring the flow by the team, in order to monitor the whole development process, occurred impacts, positive or negative. Furthermore, the measurement of lead-time and constant adjustment of the whole flow, in order to be as short as possible with aim to reduce or eliminate waste, makes the important aspect of this principle. For this, different tools and techniques are being used, such as Cumulative Flow Diagram (CFD), Control Chart, etc.

Make process policies explicit: Explicit work policies improve the collaboration inside and outside the team, by guiding their actions in certain recurring situations when team members are not supposed to decide on their own upon the further steps to be made. Making process policies explicit, the team will be empowered to handle decision with ease. Some authors [31] refer to this principle as the principle of "setting the ground rules", implying to the establishment of the common rules related to the use of the Kanban board, treatment of bugs, unexpected critical work requests, etc.

Improve collaboratively: It is needed that teams collectively discuss and reflect the workflow in order to recommend actions leading to the process improvement. In this purpose models and scientific methods are being used.

In addition to the previous Kanban software development principles stated by the pioneer in adopting the Kanban in software engineering [26], there are other principles that has been highlighted in the literature that are found important mentioning, such as [9]:

- Pulling value through the development process- referring to the implementation of pull system;
- Increasing throughput;
- Using a fixed backlog;
- Embedding quality.

IV. CONCLUSION

Nowadays, Kanban approach finds its way towards software development process. The interest in Lean Kanban approach is increasing and gaining popularity. However, despite this fact, there is no standard definition of Kanban systems for software development and no specific and unified practices and implementation methods have been defined yet. On the one hand, the reason is still not very reach and wide experience with this topic, which is, on the other hand, leading to the lack of relevant literature and case studies. Our aim was to address this issue by presenting our view and perspective on Kanban practices and its implementation in software development, based on exhausting analyses of literature and case studies on current state-of-art regarding the usage of Kanban in software engineering and our real life experience regarding Agile and Lean approach in software development.

It is concluded that adaption of Kanban with its underlying principles as a general result ensures much better understanding and cooperation between developers and stakeholders, increased productivity and work transparency, continuous flow [11], minimization of waste, maximization of customer value [10] and better understanding of the whole process [27]. Also, it is shown that these gained benefits are matching the most common motivational factors for Kanban implementation: improvement of team communication, improvement of development flow, reduced lead time, increased productivity, better transparency in organization and of work, better flow control and focus on flow and absence of fixed iterations. Furthermore, the most appropriate practices of Kanban are suggested and steps in its implementation are presented. Additionally, it was necessary to emphasis the challenges and obstacles in implementation of Kanban in these concept. Different environment form the one Kanban was developed for, lack of trainings, skills, knowledge and experiences, organizational culture and readiness for changes are main challenges every organization is facing when trying to introduce Kanban to its software development process. Even though so far Kanban has been shown as relatively beneficial for IT projects, it is needed to analyze with details the demands and nature of every project and make a decision about Kanban implementation based on those analyses.

We hope these paper helps to perceive Kanban software development more deeply and to continue further research on this topic, which we find extremely important and essencial for the future software engineering evolution.

LITERATURA

[1] The Chaos Report, The Standish Group International Inc., West Yarmouth MA, 2015.

[2] A Guide To The Project Management Body Of Knowledge (PMBOK Guide). Newtown Square, Pa. : Project Management Institute, Inc., 2004. Print.

[3] C. Nwakanma, B. Asiegbu, C. Ogbonna and P. Njoku, "Factors affecting successful implementation of information technology projects: experts' perception," European scientific journal, vol. 9, pp. 128-137, September 2013.

[4] The Chaos Report, The Standish Group International Inc., West Yarmouth MA, 2014.

[5] M. Aoyama, „Web-based agile software development," IEEE Software, vol. 15, pp. 56-65, November 1998.

[6] *Manifesto for Agile Software Development*. Retrieved from http://www.agilemanifesto.org, 2015, December 26.

[7] G. Destefanis, R. Tonelli, G. Concas and M. Marchesi, „An analysis of anti-micro-patterns effects on fault-proneness in large Java systems," Proceedings of the 27th Annual ACM Symposium on Applied Computing , Melbourne, ACM 2003, pp. 1251-1253

[8] *Scrum Alliance*. Retrieved from www.scrumalliance.org, 2015, December 28.

[9] H. Lei, F. Ganjeizadeh, P Jayachandran and P. Ozcan, "A statistical analysis of the effects of Scrum and Kanban on software development projects," Robotics and Computer Integrated Manufacturing, Available online 17 December 2015, in press, Corrected Proof.

[10] E. Corona and F. Pani, "A review of lean-kanban approaches in the software development," in WSEAS transactions on information science and applications, vol 10, January 2013.

[11] X. Wang, K. Conboy, and O. Cawley, "Leagile" software development: An experience report analysis of the application of lean approaches in agile software development," The Journal of Systems and Software, vol 85, pp. 1287– 1299, February 2012.

[12] T. Welo & G. Ringen, "Investigating Lean development practices in SE companies: A comparative study between sectors." in Procedia Computer Science, vol. 44: Conference on Systems Engineering Research, Hoboken, Stevens Institute of Technology, 2015, pp. 234-243.

[13] P. Middleton, A. Flaxel and A. Cookson, "Lean Software Management Case Study: Timberline Inc." Proceedings of the 6th International Conference XP: Extreme Programming and Agile Processes in Software Development, Sheffield 2005, pp. 1-9.

[14] T. Dybĺ and T. Dingsřyr, „What do we know about agile software development?" IEEE Software, vol 26, pp. 6–9, October 2009.

[15] C. Hibbs, S. Jewett and M. Sullivan, The Art of Lean Software Development: A Practical and Incremental Approach, O'Reilly Media, Inc. 2009.

[16] VersionOne, 9th Annual State of Agile Development Survey 2014.

[17] K. Petersen and C. Wohlin, "Software process improvement through the Lean Measurement (SPI-LEAM) method." The Journal of Systems and Software, vol. 83, pp. 1275–1287, February 2010.

[18] J. P.Womack and D.T. Jones, Lean Thinking: Banish Waste and Create Wealth in Your Corporation, New York, Simon & Schuster, 1996.

[19] R. Andersson, H. Eriksson and H. Torstensson, "Similarities and differences between TQM, six sigma and lean", The TQM Magazine, Vol. 18, pp. 282-96, 2006.

[20] M. Poppendieck, T. Poppendieck, "Lean Software Development: An Agile Toolkit." Boston, Massachusetts , USA, Addison Wesley Professional , 2003.

[21] J. Miller and O. Al-Baik, "The kanban approach, between agility and leanness: a systematic review." Empir Software Eng , vol. 20, pp. 1861-1897, december 2015.

[22] T. Ohno, Toyota Production System - beyond large-scale production, Cambridge, Productivity Press, 1988.

[23] M. G. Surendra, A.Y. Yousef and F. P. Ronal, "Flexible Kanban system," International Journal of Operations, vol. 19, pp.1065-1093, 1999.

[24] N. A.Rahman, S. M. Sharif & M. M. Esa “Lean Manufacturing Case Study with Kanban System.” in Procedia Economics and Finance, vol. 7:Proceedings of the nternational Conference on Economics and Business Research, Kedah, 2013, pp. 174-180.

[25] R. Turner, D. Ingold, J. A. Lane, R. Madachy & D. Anderson, “Effectiveness of kanban approaches in systems engineering within rapid response environments.” in New Challenges in Systems Engineering and Architecting, vol 8: Confrence on Systems Engineering Research , St. Louis, 2012, pp. 309-314.

[26] D. Anderson, Kanban-Successful Evolutionary Change for Your Technology Business, Seattle, David J. Anderson & Associates Inc., 2010.

[27] M. O. Ahmad, M. Oivo and P. Kuvaja, “Usage of Kanban in Software Companies An empirical study on motivation, benefits and challenges. “ Proceedings of the 9th International Conference on Software Engineering Advances, Nice, 2014, pp. 150-155.

[28] M. Oivo and M. O. Ahmad, “Kanban in Software Development: A Systematic Literature Review.” Proceedings of the 39th Euromicro Conference on Software Engineering and Advanced Applications, Santander, 2013, pp. 9-16.

[29] H. Kniberg and M. Skarin, Kanban and scrum – Making the most of both, InfoQ, 2010.

[30] C. Ladas. Scrumban - Essays on Kanban Systems for Lean Software Development, Seattle, Modus Cooperandi Lean, 2008.

[31] V. Mahnic, V. “Improving Software Development through Combination of Scrum and Kanban.” in Recent Advances in Computer Engineering, Communications and Information Technology”, pp. 281-288, 2014.

[32] L. D. Sienkiewicz, “Scrumban – the Kanban as an addition to Scrum software development method in a Network Organization” Business Informatics, pp 73-81, 2012.

[33] N. Nikitina and M. Kajko-Mattsson, “Developer-driven bigbang process transition from Scrum to Kanban,” Proceedings of the 9th International Conference on Software Engineering Advances, Barcelona, 2011, pp. 159-168.