ENHANCING THE AGILITY AND PERFORMANCES OF A PROJECT WITH LEAN MANUFACTURING PRACTICES

Nela Cvetković, Slobodan Morača, Miloš Jovanović,
Milovan Medojević, Bojan Lalić

This Publication has to be referred as: Cvetkovic, N[ela]; Moraca, S[lobodan]; Jovanovic, M[ilos]; Medojevic, M[ilovan] & Lalic, B[ojan] (2017). Enhancing the Agility and Performances of a Project with Lean Manufacturing Practices, Proceedings of the 28th DAAAM International Symposium, pp.0661-0670, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-11-2, ISSN 1726-9679, Vienna, Austria
DOI: 10.2507/28th.daaam.proceedings.093

Abstract

This study presents the case study on implementation of Lean Kanban manufacturing principles into the existing Agile software development methodology. The aim of this study was to introduce the adjusted Lean principles which were seen as adequate for enhancing project agility (flexibility and adaptability) and to test if these changes consequently result in improved project team performances. In order to investigate possible effects of Lean approach on improved project agility (flexibility and capacity to adapt to changes), data from interviews, daily team meetings and sprint retrospective were examined. Furthermore, team velocity, defects rates and rework rate data were used for comparison of current Agile methodology success and proposed Lean Agile method. Results show that by adding adequate Lean Kanban principles based on needs of project and team, development process was improved. Defect rate notable decreased while velocity increased, mostly caused by Lean practices enabling more realistic effort estimation and optimized work flow. Moreover, team reported that improved visual management tools facilitated communication and made it more transparent, enabling better and faster problem solving and task management.

Keywords: Agility; Lean practices; Agile Project Management; Project Performance Improvement

1. Introduction

Term *agility* implies flexibility, proactivity [1] and reactivity [2]. Concisely, agility can be interpreted as “ability to proactively or reactively adapt to change”[2]. In this context, agility has significant relevance for software development projects, where high uncertainty and frequent change requirements are mostly inevitable.

Most of the software products are specifically tailored with intents to fulfil 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 [3]. Alongside the major project constraints according to PMBOK [4]: 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 [2]. These challenges could not be overpassed with so-called traditional project management methodologies since those entail extensive planning, up-front analyses and design [5], are seen as robust, bureaucratic and process-centric.

As a response to the abovementioned challenges, Agile methodologies have been introduced to the software project management. Agile approach has 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 [3]. Agile methodologies place the emphasis on stronger communication and creation of useful product, while strict frameworks of predefined project phases and extensive documentation are of the secondary importance [6]. Nevertheless, some Agile principles were inadequate for certain project types that require higher level of flexibility. The time-boxed processes [7], exhausting work scheduling, and non-applicability on large-scale projects, represent the most important possible demerits, causing the constant pressure on team to finish its sprint work on time in any circumstances, push system and in some cases wrong effort estimations and time waste. During the evolution of agile project methodologies, Lean approach has gained significant attention, especially caused by opinion that certain Lean manufacturing principles could improve agility and leanness of a project and redeem for abovementioned possible problems with the most popular Agile methodologies.

Considering that the practice implementation of lean manufacturing principles into the Agile development process is relatively new, the available literature and case studies are still limited. The aim of this study is to introduce the adjusted Lean principles, which were seen as adequate for enhancing project agility (flexibility and adaptability), to the Agile software development project and to test if these changes consequently result in improved project team performances.

2. Theoretical background

2.1. Lean manufacturing principles in Agile project management

Lean thinking has its roots in Toyota (then called Toyota Production System) where has been created in the mid-70s. The publication of the book "The Machine that Changed the World" [8] started the diffusion of lean manufacturing practices first of all in automotive industry and later in all other industries from agriculture to aerospace [9]. Lean has been recognized as the mainly operational management strategy in manufacturing sector [10]. The basic principle of lean is responsiveness to change and waste minimization [11].

Recently the agile community has started to look toward Lean software development approaches [12], assaying the usage and benefits of Lean principles known for manufacturing and production industry on software development process. According to [13] there are claims that Lean software development provides the theory behind agile practices [12]. Many authors and practitioners see Lean software development as an instance of agile methods [14]. 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.

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. 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 [15].

Lean concept involves three main elements: lean concepts, lean principles and lean practices [12]. 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". 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 [16].

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" [12], [13].

Poppendieck & Poppendieck [13] 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.

Furthermore, other authors widened the list of lean practices relevant for software development [12]:

- Address bottlenecks;
- Defer decision making;
- Develop rewarding system;
- Workload levelling;
- 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 [17], [18].

2.2. Kanban

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. Kanban (kahn-bahn) is a Japanese word literally meaning “visible record” [19]. 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 [20]. 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 [20], development of flexible work stations, to minimization of waiting time and logistics costs [19]. 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”. 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.

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”[18]. 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 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 [21]. 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.

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 [22].

In their study [23] 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.

3. Case study and results

3.1. Methodology

The aim of this study was to introduce the adjusted Lean principles which were seen as adequate for enhancing project agility (flexibility and adaptability) and to test if these changes consequently result in improved project team performances. In order to analyse and compare the project productivity and effectiveness before and after Lean principles were implemented in Agile software development process, available metrics were explored and adequate metrics have been chosen.

Based on extensive literature review and project methodology characteristics the most relevant and used metrics were selected for this study:

- *Team velocity*: Represents the amount of work accomplished in each Sprint expressed in story points [24], [25], [26]. This metric demonstrates how much work development team can accomplish within one iteration.
- *Defects rate (count)*: can be both, internal process measure of software testing and external measure as the number of defects experienced by the customer [26], [27], [28].
- *Rework items*: count of items demanding follow-on efforts.

Firstly, the existing Agile Scrum method used by teams was analysed based on previous 9 iterations. Following, the main challenges that team members are facing are identified and used as input to define the directions towards improved project agility and performances.

Based on the desired improvements, adequate modified Lean practices were suggested and integrated into the Agile Scrum methodology. New software development process was tested during 10 iterations and team performance in both cases was compared using chosen metrics. Additionally, data from sprint retrospectives, supporting programs and notes from interview were analysed in order to get insight into the project members' experience.

3.2. Analysis of applied Agile software development methodology

3.2.1. Agile Scrum software development process applied with the project team

The goal of development process analysis was to get insights into the software project management and identify improvement possibilities for this project. Development team has 7 collocated sub teams, counting all together 30 project team members. The final results of the projects was 3D computer game. The team was using adjusted Agile Scrum methodology, applying Scrum practices and artefacts presented in table 1.

Existing Scrum practices within project team	
Product backlog	In this case backlog product is "Game design document" which contains all game chapters with their rough description. The backlog items are prioritized from the beginning of the project.
Sprint and sprint backlog	There are two-week time-boxed iterations. Sprint backlog items are elements and objects of certain game scene. During the sprint duration, each member executes sprint items which go through different process phases.
Sprint planning and effort estimation	There is no traditional sprint planning. Since sprint backlog is made of elements and objects of one scene (small user stories), there is no need for sprint items planning. However, the team is revising the items together in order to discuss if the given effort estimation is realistic
Regular virtual board and burn down charts	For monitoring project progress Trello program is being used with integrated virtual table. Also, the program provides team members with burndown charts showing remained work within iteration. Overall, these artefacts are helping team members and product owner to track the progress of the sprint.
Scrum meetings	As the goal have discussion about necessary corrections, current problems and further instructions.
Sprint retrospective	Scrum retrospective helps team improve their further iterations. It is being conducted with aim to constantly improve following sprints.
Sprint demo	It represents the official representation to the product owner of functional sprint increment and official approval of sprint increment.

Table 1. Existing Agile Scrum practices and artefacts in case company

Following, the described development process with Scrum methodology is presented graphically (figure 1).

Fig. 1. Graphical presentation of Agile Scrum software development process applied with the project team

3.2.2. Team velocity analysis

Velocity was analysed and presented for each subteam and for the whole development team. In this paper the planned and real velocity of the whole development team is shown in chart 1.

Chart 1. The planned and real team velocity

Results demonstrate high number of unfinished story points per iteration. With goal to identify more precisely potential problems within each sub team, planned and expected velocity was additionally analysed for each sub team separately. The major problem was in one sub team where members could not accomplish even 50 percentage of planned work. These events and delays in this particular sub team and similar problems in other sub teams caused serious bottlenecks, since all unfinished project tasks from one sprint need to be reallocated to the next iterations. Furthermore, one of the reasons for appearance of bottlenecks were interdependencies among tasks performed by different sub teams. Team members defined the unrealistic work load as important cause of poor velocity.

3.2.3. Analysis of defects and rework items

During the development process finished increments are regularly tested manually and automatically, in order to determine the quality of performed tasks and to detect necessary rework and possible improvements. During the project duration project team has developed bug prediction practice. Team is forecasting possible defects that will demand additional human effort reducing the chances for bottlenecks. However, analysis show many of occurred

defects were not among predicted once, enhancing the lack of experience and appropriate methods for risk assessment regarding bugs and defects. Consequently, during iterations few more complex tasks were blocked. The execution of those tasks has been stopped in order to continue with the iteration which has resulted in accumulation of unfinished sprint items.

To get deeper insight on team's performance, defects count and rework items are demonstrated in chart 2. The number of defects and rework items is represented for each of the subteams.

Chart 2. The average number of defects and rework items per each team

The highest number of bugs and needed rework is identified in Level design team. The main cause is the intensive communication Level design team had to have with 3D team members and Programming team, resulting in lack of time for execution of tasks and inefficient communication.

3.2.4. Analysis of sprint retrospective data

Alongside the examination of velocity and defects rate as indicators of project productivity and efficiency, data from sprint retrospectives, meetings and project management programs were analysed and the main challenges project team is facing during the Scrum development process are defined:

- *Incomplete tasks within a sprint* caused by work overload of team members, inflexible sprint duration (“time-boxed” iterations) and complex sprint items (tasks). Furthermore, since Scrum methodology do not allow changes in sprint backlog, change requirements occurred during the iteration remain unperformed regardless their priority, or, in case they are integrated in sprint backlog, other tasks are not being postponed, again resulting in incomplete tasks within sprint. Finally, there is lack of process transparency among sub teams which is essential in this case where tasks are highly interdependent. Even though time-boxed iterations can be motivational for team members to perform their tasks on time, in this case this practice showed lack of benefits for project agility.
- *Poor effort estimation for certain user stories or its parts* resulted in unrealistic scheduling, causing either idle time or bottlenecks.
- *Frequent change requirements during the sprint that team could not fulfil;*
- *Idle time for some team members:* Team reported that certain sprints were not optimized, respectively, work load among sub teams was not balanced enough. Consequently, not rarely some team members were required to wait for other team members to finish their task in order to continue their work. Since sprint backlog is locked, team member waiting for the upcoming task was not able to start new task, which resulted in idle time.
- *Bottlenecks:* During the iterations high number of sprint items were blocked in certain implementation phase. When certain problem occurred, team members were most commonly postponing blocked items and started new once without resolving issue, mostly because they were impeded to dedicated additional time to certain task due to the fixed iteration. In addition, poor effort estimation inflicted stoppage of certain work items.

3.3. Identification of improvement directions

In the next phase of this study, adequate directions for project performances improvement were defined. Observing identified challenges project team is facing and summarizing abovementioned problems, it was concluded that agility of the project needs to be improved in order to obtain better project results. Project needs more flexibility in order to successfully meet change requirements, better process optimization in order to mitigate bottlenecks and idle time and more efficient visual management. Main development process improvement directions proposed were:

- *Improvement of work flow optimization:* Idle time and work overload could be mitigated by achieving smoother development process, respectively, by improving work flow optimization. In order to accomplish that, it is necessary to plan work more efficiently, to have more realistic effort and time estimation for each work item and clear instructions in case of blocked tasks.
- *Improvement of efficiency metrics based on experience from previous iterations:* Although it was observed that team members are not able to finish the backlog items within sprints, sprint planning method and velocity estimation was not changes during first 9 iterations. The major cause for delays on project were unrealistic effort estimation and complex work items. Thus, it is essential to dedicate attention to examination of previous iterations and performance results in order to obtain certain time norms and standards to tasks of approximate complexity.
- *Improvement of work flow transparency and team members' communication:* Blocked items and delays were also the result of inadequate cooperation and communication mechanisms. In addition, insufficient insight into the sprint progress of other sub teams caused inability for some team members to plan their related work properly.

3.4. Integration of Lean principles into the Agile project management methodology

Appropriately applied Lean development improves efficiency in eliminating waste and increases efficiency in delivering value to the customers [6]. In this case, it means Lean practices can provide project team with possibility to detect which change requirements are important, to prioritize them and include in their development process. Additionally, pull system characteristic for Lean approach has proved to be essential for process flow optimization, allowing the team to produce and deliver what customer wants while mitigating bottlenecks. The main overall goal of Lean development is to achieve continuous and smooth flow of production, maximizing flexibility [12].

According to outlined above and previous chapter, identified challenges and defined improvement directions, adequate Lean Kanban principles are suggested to be implemented and combined with existing (figure 2):

- Constant backlog planning and prioritization;
- Introduction of „pull“ system;
- Introduction of „ready“ queue;
- Work items of similar complexity;
- WIP (work in progress) limits;
- Improved work flow visualization;

Fig. 2. Lean development principles suggested based on noted challenges and defined improvement directions

Constant backlog planning and prioritization: In Kanban Lean software development the sprint backlog is not locked. The backlog consists of items prioritized and chosen by creative director and game designer. This way, changes during the sprint are acceptable and idle time avoided, since as soon as a team member finishes one work, in case there is free capacity, employee pulls next items and works on it.

From "push" to "pull": The system implies that work is pulled as and when needed into a queue, unlike the approach used in a traditional Scrum where all the work to be completed within a sprint is assigned in the beginning of the sprint to the sprint backlog.

Introduction of „ready“ queue: This ready queue contains items that are pending from the backlog, but have high priority and are chosen by product owner.

Work items of similar complexity: The suggestion is to decompose all of the work items on tasks of similar sizes as much as possible. This way during the following iterations it is possible to establish approximate time standards for familiar

tasks. Furthermore, team members will have more clear insight into the work scope and will be able to improve effort estimation.

WIP (work in progress) limits: This principle implies limiting the Product Backlog Items (PBIs) that are in progress at any point of time, including the sprint backlog. The idea is to keep team focused on completing work at hand rather than starting a new task. This means once a limit is reached within a particular stage of the workflow, rather than starting working on something new, it is time to help someone else within the team. This will ensure that the team's work flow becomes smoother and no stage becomes a bottleneck.

3.5. Results

The effects of proposed Lean Kanban principles on team performance were measured during 10 two-week iterations. During iterations before Lean principles were introduced, team was defining velocity as team capacity to perform certain number of story points within one sprint. Those story points weighted determined coefficient representing effort necessary for completing the task, while each item weighted certain number of those story points. In such a way, for every sub team expected velocity, i.e. number of story points is planned in accordance with the number of sub team members and work complexity. However, sprint items could comprise significantly different story points. In the case of Lean development, work items contain same number of story points since one of the suggestions was to reduce work items to approximately same-size items.

It is noticed that modified agile methodology has resulted with the first two iterations with low team velocity. However, after the third sprint team accomplished significantly better velocity than with previous methodology. According to team members, the velocity of the first three iterations is the result of changes and new working practices which have required team members to adapt and learn. Planned and real velocity of the whole development team is shown in chart 3.

Chart 3. The planned and real team velocity after introduction of Lean principles

As in the case of initial software development process, defects and needed rework were monitored. The status of defects count after 10 iterations is shown in chart 4.

Chart 4. The average number of defects and rework items per each team after introduction of Lean practices

Although relatively high in first iterations, after the third sprint the number of defects and items demanded to be reworked has been significantly lower than in the first case. Team has shown more noticeable improvement regarding work quality compared to velocity melioration. Defect and rework rate was reduced for even 50 percentage. Team members emphasized that this results was mainly caused by decreased pressure due to the implementation of constant backlog grooming practice and work in progress limitation.

Furthermore, during interviews and sprint retrospectives, team members pointed out that adequate usage of metrics and visual management tools facilitated communication and made it more transparent, enabling better and faster problem solving and task management. Rework was reduced, since the quality of work performed was monitored and controlled during the whole sprint not only during the demo sessions. In addition, team was able to accomplish more user stories and improve its productivity, since metrics enabled more realistic estimation of work items planned for one sprint.

4. Discussion and conclusion

The aim of this study was to test the influence of Lean practices on project performances and project agility, i.e. flexibility and potential to adapt to changes.

The main challenges team using Agile Scrum approach is facing were identified and based on results suggestions for the improvement were created.

Scrum methodology, as other agile methods, is not being implemented in the same manner in every organization. In opposite, each organization adjusts methodology based on their specificities, project and team characteristics. Since the product of this project is alpha game, constant feedback from publisher and customers are included during the whole project duration. Thus, change requirements were highly frequent. Scrum methodology implies that sprint backlog could not be changed during the iteration, team was unable to fulfil new requirements on time. Consequently, the project milestones were often not accomplished, postponing the finish of the project. Furthermore, team consists of very different profiles of employees (artists, programmers, philosophers, etc.), which makes development process management more complicated and demands different techniques for effort management.

By adding adequate Lean Kanban principles based on needs of project and team, game development process was improved.

Pull system and constant backlog grooming enabled team to be responsive to the change requirements without causing delays and unfinished items within sprint.

WIP limits ensured that team members are focused on tasks they are executing at the moment and on solving potential problems and blocked items immediately, consequently decreasing the occurrences of bottlenecks.

The practice of decomposing sprint items into similar-size tasks improved effort estimation process and facilitated the sprint progress monitoring. Additionally, it enabled team to establish average lead times for future iterations.

Considering the company, project nature, team structure, and obtained results, further suggestion from authors is to completely eliminate sprints in order to optimize the flow additionally and improve capacity to react fast to the change requirements. Eliminating time-boxed iterations the optimization of work flow would be even more improved due to more flexible and event-driven work planning. Responsible project leader or product owner should continue grooming backlog and prioritizing items.

5. References

- [1] Naylor, J. Ben, Naim, M. M., & Berry, D. (1999). Leagility: Integrating the lean and agile manufacturing paradigms in the total supply chain. *Int. J. Production Economics*, Vol.62, 107–118.
- [2] Stratton, R., & Warburton, R. D. (2003). The strategic integration of agile and lean supply. *International Journal of Production Economics*, Vol. 85, No.2, 183–198
- [3] The Standish Group International Inc. (2015). The Chaos Report. Available from <https://www.projectsmart.co.uk/white-papers/chaos-report.pdf>. Accessed: 01.09.2017.
- [4] A Guide To The Project Management Body Of Knowledge (PMBOK Guide) (2004). Newtown Square, Project Management Institute, Inc., 2004. Print.
- [5] Destefanis, G., Tonelli, R., Concas, G., & Marchesi, M. (2013). An Analysis of Anti-Micro-Patterns Effects on Fault-Proneess in Large Java Systems. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing* (pp. 1251–1253).
- [6] Majstorovic, V., Polajnar, A. & Bandic-Glavas, M., (2015). Innovative PM Approach to Produce an Environmentally Friendly Product, Chapter 36 in *DAAAM International Scientific Book 2015*, pp.423-430, B. Katalinic (Ed.), Published by DAAAM International, ISBN 978-3-902734-05-1, ISN 1726, Vienna, Austria, DOI: 10.2507/daaam.scibook.2015.36
- [7] Aoyama, M. (1998). Web-based agile software development, *IEEE Software*, vol. 15, pp. 56-65.
- [8] Herzog, V., Medic, M. & Kostanjevac, T., (2008). Value Stream mapping for Effective Lean Manufacturing, *Annals of DAAAM & Proceedings*, B. Katalinic (Ed.), Published by DAAAM International, ISN 1726-9679, Vienna, Austria

- [9] Womack, J.P.; Jones, D.T. & Ross, D. (1990), *The Machine that changed the World*. Macmillan Publishing Company, Canada.
- [10] Welo, T., & Ringen, G. (2015). Investigating Lean Development Practices in SE Companies: A Comparative Study Between Sectors. *Procedia Computer Science*, Vol. 44, 234–243
- [11] Vujica Herzog, N & Buchmeister, B. (2012). Lean Manufacturing in Slovenian Companies, Chapter 11 in *DAAAM International Scientific Book 2012*, pp. 117-128, B. Katalinic (Ed.), Published by DAAAM International ISBN 978-3-901509- 86-5, ISSN 1726-9687.Vienna, Austria
- [12] Wang, X., Conboy, K. & Cawley, O. (2012). “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.
- [13] Poppendieck, M. & Poppendieck, T. (2003) *"Lean Software Development: An Agile Toolkit."* Boston, Massachusetts, USA, Addison Wesley Professional.
- [14] Dybl, T. & Dingsr, T. (2009) „What do we know about agile software development?“ *IEEE Software*, vol. 26, pp. 6–9.
- [15] Petersen, K. & Wohlin, C. (2010). “Software process improvement through the Lean Measurement (SPI-LEAM) method.” *The Journal of Systems and Software*, vol. 83, pp. 1275–1287.
- [16] Andersson, R., Eriksson, H. & Torstensson, H. (2006) “Similarities and differences between TQM, six sigma and lean”, *The TQM Magazine*, Vol. 18, pp. 282-96.
- [17] Hibbs, C., Jewett, S& Sullivan, M. (2009). *The Art of Lean Software Development: A Practical and Incremental Approach*, O'Reilly Media, Inc.
- [18] Miller, J. & Al-Baik, O. (2015) "The kanban approach, between agility and leanness: a systematic review." *Empir. Software Eng* , vol. 20, pp. 1861-1897.
- [19] Ohno, T. (1988). *Toyota Production System - beyond large-scale production*, Cambridge, Productivity Press.
- [20] Rahman, N. A., Sharif S.M. & Esa, M. M. (2013) “Lean Manufacturing Case Study with Kanban System.” in *Procedia Economics and Finance*, vol. 7:Proceedings of the international Conference on Economics and Business Research, Kedah, pp. 174-180.
- [21] Oivo, M. & Ahmad, M. O. (2013) “Kanban in Software Development: A Systematic Literature Review.” *Proceedings of the 39th Euromicro Conference on Software Engineering and Advanced Applications*, Santander, pp. 9-16.
- [22] Kniberg, H. & Skarin, M. (2010). Kanban and scrum – Making the most of both, InfoQ.
- [23] Ahmad, M. O., Oivo, M. & Kuvaja, P. (2014) “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.
- [24] Mahnic, V., Zabkar, N. (2012) *Measuring Progress of Scrum-based Software Projects*. *Elektronika ir Elektrotechnika*, vol. 18, pp. 73-76.
- [25] Nwakanma, C., Asiegbu, B., Ogbonna, C. and Njoku, P. (2013) Factors affecting successful implementation of information technology projects: experts’ perception. *European scientific journal*, vol. 9, pp. 128-137.
- [26] Kupianen E. et Al. (2015). Using metrics in Agile and Lean Software Development. *Information and Software Technology*, vol. 62, pp. 143-163.
- [27] Talby, D., Hazzan, O., Dubinsky, Y. and Keren A. (2006). Reactions on reaction in agile software development. In *Proceedings - AGILE Conference 2006*, vol. 2006, pp. 100-110.
- [28] Green P. (2011). Measuring the impact of scrum on product development at adobe systems. In *Proceedings of the Annual Hawaii International Conference on System Sciences*.