Improving Software Development through Combination of Scrum and Kanban

VILJAN MAHNIC
Faculty of Computer and Information Science
University of Ljubljana
Trzaska 25, SI-1000 Ljubljana
SLOVENIA
viljan.mahnic@fri.uni-lj.si

Abstract: - Applying Kanban principles to software development is becoming an important issue in the area of software engineering. The number of Scrumban users, a method that combines Scrum and Kanban practices, almost doubled in the last year. While Scrum is well-known and widespread method, scientific literature about Kanban adoption is scarce and many questions regarding Kanban use in practice (e.g., the structure of Kanban board, setting appropriate work in progress limits, combining Kanban and other agile methods, etc.) are still open. The aim of this paper is to upgrade previous research of different Kanban boards by proposing structure of a generic Kanban board and ground rules for its use, which could serve as a guiding reference for teams planning to introduce Scrumban as their development process. Additionally, an example of using such a board in practice is provided. The paper starts from the premise that appropriate combination of Scrum and Kanban advantages makes the most from both. Therefore, the proposed solution combines the most important Scrum practices (release and Sprint planning, regular delivery of increments, frequent feedback) with basic Kanban principles (visualization of workflow, limiting work in progress, change management).

Key-Words: - Scrum, Kanban, agile methods, software development, software process management, quality improvement

1 Introduction

Numerous agile methods [1] have appeared in the last fifteen years that – in contrast to traditional disciplined approach – value individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [2]. Several reports indicate that agile teams achieve significant improvements in productivity, quality, and stakeholder satisfaction, e.g., [3, 4, 5]. According to Standish Group's 2011 Chaos report, the success rate for agile projects is 42%, which is three times more than for waterfall projects (14%) [6]. In spite of initial doubts (some people even considered agile methods a backlash to software engineering [7]), agile methods have been gaining wide acceptance among mainstream software developers. In January 2010 Forrester [8] reported results of their Global Developer Technographics Survey, which revealed that 35% of respondents used an agile, 21% an iterative, and 13% a waterfall development process, while 31% did not use a formal process methodology.

According to the last State of Agile Development Survey [9], the most widespread agile method is Scrum [10, 11]. Pure Scrum is reported to be used by 54%, Scrum/XP Hybrid by 11%, and Scrumban by 7% of respondents. However, the survey also revealed a rapid growth of the number of Kanban users. Compared to 2011, Kanban and Kanban variants [12, 13] nearly doubled in 2012, mostly due to an uptick in Scrumban use. It is expected that this trend will continue in the forthcoming years; therefore, applying Kanban approach to software development (either as a standalone method or in combination with Scrum) seems to be a hot topic for software researchers and practitioners.

While Scrum advocates incremental software development through a sequence of iterations, the main goal of Kanban is to maximize the workflow and shorten the lead time (i.e., the average time to complete one item) by limiting the amount of work in progress (WIP). In order to visualize the workflow, a Kanban board is used, which consists of several columns, each column representing a stage in the development process. The number of work items in each column is limited, thus forcing the developers to concentrate on the work in

progress and not start a new piece of work until the previous one is completed.

Kanban has been used in manufacturing for more than two decades [14], but it is a relatively new concept in the area of software engineering. A recent review of Lean-Kanban approaches to software development [15] revealed that there was almost no paper published on this topic in the scientific literature. However, some early adopters report significant improvements in their development process. Sjøberg et al. [16] describe the case of a Scandinavian company, which after the introduction of Kanban almost halved the lead time, reduced the number of weighted bugs by 10 percent, and improved productivity. Similarly, Anderson et al. [17] report experience from a Microsoft's maintenance project indicating that the typical lead times, from the arrival of a request to its completion, reduced from 125–155 days to only 14 days after successful implementation of Kanban. Some other papers explore elimination of waste [18, 19] and describe introduction of Kanban in combination with iterative development [20, 21]. Nevertheless, it can be concluded that the current state of research on Kanban use in the area of software engineering is still in the nascent phase.

Due to limited evidence there are still many open questions regarding Kanban implementation in practice, e.g., how the Kanban board should like, how to set the WIP limits, how to combine Kanban with other agile practices and methods, what tools are available for managing the Kanban process, etc. Corona and Pani [15] studied 14 different Kanban boards in order to identify their main characteristics and main activities defining the software process. The aim of this paper is to upgrade their research by (1) proposing structure of a generic Kanban board, suitable for smooth transition from Scrum to Scrumban, (2) describing ground rules for maintaining such a board, and (3) providing an example of its use in practice. By generic we mean a Kanban board that includes a full range of activities that make up a software process, and can serve Scrumban users as a guiding reference when adapting the board structure to their needs.

We start from the premise that iterative development and Kanban are not mutually exclusive but complementary processes that when used together contribute to higher performance of agile teams [20]. Therefore, combining Scrum and Kanban practices can contribute to make the most of both [22] and represents the first step in introduction of lean principles to software development as proposed by Ladas [13].

While some Kanban enthusiasts treat particular Scrum practices (e.g., effort estimation, Sprint planning meetings) as waste, we agree with those (e.g., [23]) who argue that some up-front planning is necessary (or even inevitable) in development projects (when a new product is developed from scratch and the development team is faced with a comprehensive backlog), and can only be omitted in pure maintenance projects (bugs and small enhancements when work requests arrive spontaneously from time to time) if the team obtains enough experience with Kanban. Therefore, the solution proposed in this paper can be regarded as an example of Scrumban – a software development method that combines practices of Scrum and Kanban. This method is particularly appropriate for companies already using Scrum and wishing to make their development process leaner, even more transparent, and continuously optimizing.

In order to incorporate best practices of Scrum and Kanban, the advantages of both methods are analyzed first (Sections 2 and 3). Then the proposed generic Kanban board and ground rules are described (Section 4). An example of adaptation of the proposed board to the needs of a small team (including the choice of appropriate WIP limits) is provided in Section 5. Finally, the most important conclusions are summarized in Section 6.

2 Scrum Advantages

Scrum advantages that we want to retain in the new Scrumban approach comprise project planning on the basis of fix-length iterations (called Sprints), regular delivery of product increments, and frequent feedback on work performed. It is assumed that the development process is driven by the Product Backlog – a prioritized set of user requirements formulated as user stories [24]. Each requirement is recorded as a user story consisting of three parts: a written description (used for planning and as a reminder), conversations about the story (to flesh out the details), and acceptance tests (to determine when a story is "done").

2.1 Project Planning

The last State of Agile Development Survey [9] revealed that the lack of up-front planning and loss of management control are the greatest concerns about adopting agile. Therefore, when introducing Kanban it makes sense (at least at the beginning) to retain well-established Scrum practices of effort estimation, and release and Sprint planning [24, 25].

Scrum requires that the effort for implementation of each user story is estimated in story points, and the stories comprising the Product Backlog are allocated to Sprints strictly considering estimated velocity of the development team (i.e., the number of story points the team can accomplish in one Sprint). In such a way, a release plan is created that makes it possible to define approximate release date and serves as a guidepost toward which the project team can progress.

At the beginning of each Sprint the development team re-estimates its velocity and the remaining user stories in order to obtain a more precise Sprint plan, called Sprint Backlog. The content of the Sprint Backlog is defined in co-operation with the Product Owner at the Sprint planning meeting. The Sprint Backlog consists of stories with highest priority having the total number of story points equal to the Team's estimated velocity. The Team further decomposes each story into constituent tasks and assigns responsibility for each task. Each team member individually estimates how many hours it will take to accomplish each task he/she accepted.

2.2 Regular Delivery of Increments

Each Sprint should produce a visible, usable, and deliverable increment of functionality. Since a Sprint is time-boxed development (i. e., the end date for a Sprint does not change), developers are motivated to actually finish stuff and release it instead of having huge piles of 99% finished user stories. On the other hand, customers see on-time delivery of increments. They receive tangible results in regular intervals, thus having an opportunity to see how the product actually works as soon as possible. Through regular delivery of increments a better relationship with the customers develops, trust builds, and mutual knowledge grows.

2.3 Frequent Feedback

The system of Sprints encourages regular feedback. Each Sprint is followed by a Sprint review, which provides an opportunity for all stakeholders to reflect on the results of previous Sprint and discuss improvements for the next. Stakeholders are free to voice their comments, observations or criticism regarding results.

Due to constant feedback, it becomes easier to cope with the changes and the work done better meets the customers' needs. New features can be added and tasks reprioritized without adversely affecting the project flow, so that progress is made even when requirements are not stable.

3 Kanban Advantages

Kanban has become popular because of its ease of implementation, use of visual controls, work in progress management, and relentless focus on the continuous process improvement.

3.1 Limiting Work in Progress

Kanban is based on a very simple idea: Work in Progress (WIP) should be limited and something new should be started only when an existing piece of work is delivered or pulled by a downstream function. This mechanism is known as a "pull" system: new work is "pulled" into the system when there is capacity to handle it, rather than being "pushed" into the system from the outside. WIP limit defines the capacity of each step in the development process in terms of the number of work items that may be in progress at each workflow state.

Appropriate WIP limits ensure that a pull system cannot be overloaded, thus making it possible to maintain a sustainable pace of development. Only the workers at the bottleneck are fully loaded; everyone else should experience some slack time. On the other hand, WIP limits quickly bring 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.

Limiting WIP significantly reduces lead time, which is used as a major measure of development team throughput and productivity.

3.2 Process Visualization

Kanban requires work items (usually user stories) to be presented on a Kanban board, which serves as a visual control mechanism indicating how the work flows through the various stages of development process. Typically, a whiteboard with sticky notes, or an electronic card wall system is used.

The Kanban board consists of a sequence of columns that represent the various states a work item can exist in during the development process. As work progresses through the development lifecycle, the cards move from one state to the other, until they finish in the last column. Each column has on its top a WIP limit indicating how many cards can be in the corresponding workflow state at any one time. When a card is completed in one column, it moves to the next, thus creating an open space in

its current column, which allows the development team to pull a completed card from a previous column. If, for any reason, cards in one column cannot be completed and moved forward, this column sooner or later hits its WIP limit, which prompts the development team to fix the bottleneck instead of just piling up a whole bunch of unfinished work.

3.3 Change Management

By providing team members and other stakeholders with visibility into bottlenecks and their impact on the development process, Kanban encourages collaboration among all parties involved and discussions about improvements, thus contributing to incremental evolution of existing processes and continuous improvement. As such, Kanban represents an approach to introducing change to an existing software development lifecycle or project management methodology. Kanban does not require revolutionary changes, but can be simply incorporated into an existing development process. It is only necessary to visualize the workflow, limit WIP, and measure the lead time.

4 Generic Kanban Board

The aim of the generic Kanban board shown in Fig. 1 is to provide a full range of activities (steps in the development process) and approaches to board modeling (working states, buffers) that must be taken into account when deciding about the most appropriate board structure. Although the proposed board can be used as such, it is primarily intended as a reference from which the users can choose those parts that best suit their development process.

4.1 Columns Description

The proposed Kanban board consists of the following columns:

- The **"Product Backlog"** column contains all user stories currently known. Each user story is prioritized by the Product Owner and estimated using planning poker [26, 27]. Whenever a new user story is created, a new card is added to this column.
- The **"Sprint Backlog"** column contains user stories belonging to the current Sprint. The content of this column is initiated at the Sprint planning meeting when the Product Owner and the development team agree which stories to develop in the next Sprint. During Sprint, the stories move to subsequent columns in accordance to Kanban pull mechanism. If the Sprint is executed properly, this column becomes empty at the end of the Sprint and is filled again at the next Sprint planning meeting.
- The **"Next"** column is intended to contain a limited number of high priority stories. Whenever a member of development team is ready to start working on a new item, he/she can take a user story from the "Next" column and move it to "Analysis & Design".
- The **"Analysis & Design"** column reflects Scrum approach to just-in-time design requiring that each user story is decomposed into constituent tasks. Given the fact that user stories provide only rough description of required functionality, it is also assumed that all details regarding implementation are clarified during this step. The "Analysis & Design" column is split into two sub-columns "Ongoing" and "Done" in order to distinguish between user stories being still in a working state ("Ongoing") and user stories for which analysis and design were completed ("Done"). In such a way, the "Done" sub-column acts as an inter-process buffer providing information which user stories can be pulled to the next step in process. The same holds for "Ongoing" and "Done" sub-columns within "Development", "Testing", and "Documentation".
- The **"Development"** column is intended to show what is (at a given moment) being developed. In our case it is assumed that development includes coding and unit testing performed by programmers; therefore, development is considered "Done" when all unit tests pass and functional testing can start.
- The **"Testing"** column indicates which user stories are being tested. It is assumed that this step comprises functional and integration tests performed by the development team in order to verify that user stories comply with the Scrum concept of done, which requires that each Sprint provides a fully shippable increment of product functionality. Stories in the "Done" sub-column are considered completed and ready to be documented.
- The **"Documentation"** column contains user stories for which documentation is being prepared. Scrum requires that the user operation of the new functionality is documented, either in Help files or in user documentation [11]. This requirement is a constituent part of the definition of done. Stories in the "Done" sub-column are considered completed and ready to

be presented to the Product Owner and other stakeholders at the Sprint review meeting.

- The **"Acceptance"** column corresponds to final acceptance testing performed by the Product Owner. According to Scrum rules, it is the Product Owner who decides whether a user story implementation meets all requirements and can be put in production.
- The **"Deploy"** column shows which user stories are being deployed.
- The final **"Done/Live"** column contains stories that were accepted by the Product Owner ("Done") and made available for use ("Live").

4.2 Ground Rules

In Kanban it is recommended to establish ground rules for who uses the board and how [22]. For each column and subcolumn it should be specified who can move a card in (from previous column), and who can move a card back (to one of the previous columns). Special attention should be devoted to treatment of bugs and unexpected critical work requests.

Considering the standard Scrum roles of Product Owner, ScrumMaster and development team, an example of possible ground rules for Kanban board in Fig. 1 is provided:

- A card can be created in the "Product Backlog" only by the Product Owner.

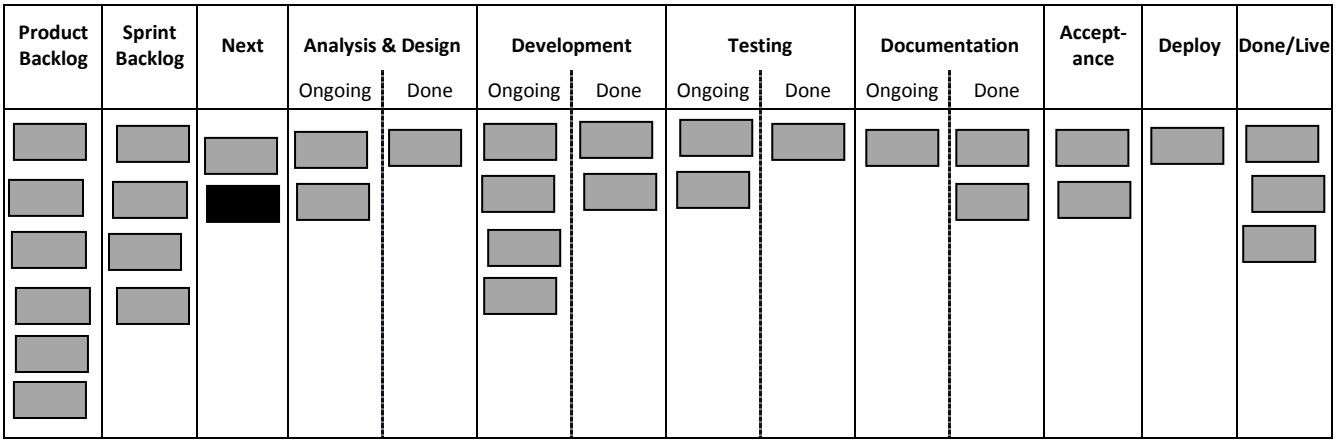

Fig. 1 Proposed generic Kanban board

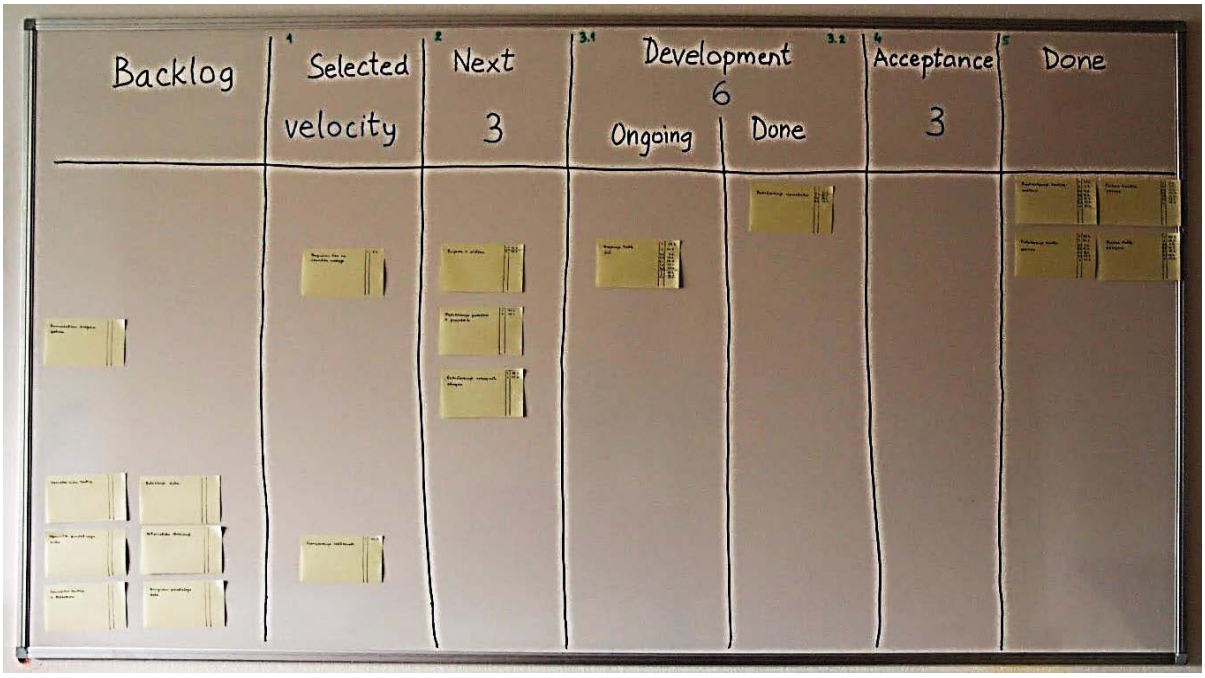

Fig. 2 Example of an adapted Kanban board described in Section 5

- Cards belonging to user stories that are to be developed in the next Sprint are transferred from the "Product Backlog" to "Sprint Backlog" by the Product Owner or ScrumMaster. This operation takes place at the Sprint planning meeting. Strictly following Scrum rules, moving a card from Product Backlog to Sprint Backlog and vice versa is prohibited in the middle of a Sprint (unless agreed otherwise).
- A card can be moved to the "Next" column only by the Product Owner. When a user story is taken from the "Next" column (in order to be developed), the Product Owner has to choose next story from the "Sprint Backlog" with the highest priority and fill up the free space in "Next". Within the WIP limit of the "Next" column, the Product Owner is allowed to change priorities by moving high priority stories from the "Sprint Backlog" to "Next" and vice versa. Whenever the number of user stories reaches the WIP limit, one of them had to be moved back to the "Sprint Backlog" before being replaced with a more urgent user story.
- Members of development team are allowed to make changes in the "Analysis & Design", "Development", "Testing", and "Documentation" columns (as well as corresponding sub-columns). Whenever one of them is ready to start working on a new item, he/she can take a user story from the "Next" column and move it to the "Analysis & Design" column. The story then moves through "Development", "Testing", and "Documentation" until it reaches the "Done" sub-column in "Documentation", which indicates that the story is fully tested, documented and integrated in the shippable increment of new functionality.
- The Product Owner can take a story from the "Documentation Done" sub-column and start its evaluation within "Acceptance". If the story is accepted, he/she moves the corresponding card to the "Deploy" column (if deployment is organized as a separate activity) or the final "Done/Live" column (in case or small projects or if deployment is not treated as a part of development process).
- The "Deploy" column can be used by a separate deployment team, which (after successfully deploying the new increment of functionality) moves the corresponding cards to the final "Done/Live" column.
- Serious bugs and unexpected critical requests are marked with a different color (see the black card in Fig. 1), and are treated as "silver bullets" entering directly the "Next" column as proposed by Ladas [13, pp. 172-174]. A silver bullet still has to go through the workflow and obey the WIP limits in order not to interrupt work-in-process.

Note, however, that the above ground rules are just an example of possible ground rules. Each team is advised to set its own rules and experiment with them to achieve best fit with their development process.

5 Adaptation for a Small Team

The Kanban board in Fig. 1 is suitable for complex development projects, but in most cases it should be adapted to serve specific needs of a particular team and/or project. In such cases it can be used as a guiding reference containing a list of possible working states and ground rules.

In this section a real-life example is provided of how a subset of proposed columns was used and how WIP limits were set in an experimental Kanban project that was conducted at the Faculty of Computer and Information Science, University of Ljubljana. The project required the development of a web-based tool for managing Kanban projects and lasted 3.5 months. The development team consisted of three graduate students of Computer Science, while the author played the role of the Product Owner. During the project we experimented with two different boards and different WIP limits [28]. The final board structure and WIP limits are shown in Fig. 2.

The board consisted of a subset of columns proposed in Fig. 1. The "Backlog" and "Selected" columns correspond to the "Product Backlog" and "Sprint Backlog", respectively.

The WIP limit of the "Selected" column was expressed in terms of velocity, i.e., the number of story points that the team was expected to complete in a Sprint. At each Sprint planning meeting the corresponding stories were transferred form the "Backlog" column to "Selected".

The WIP limits of other columns were expressed in terms of work items, indicating the maximal number of user story cards in each column. The "Next" column contained a limited number of high priority stories as described in Section 4. The WIP limit was set to 3 because there were 3 developers working on the project.

The project was small and each developer was supposed to develop a user story from beginning to end; therefore, it seemed reasonable to merge the "Analysis & Design", "Development", "Testing", and "Documentation" columns into a single

"Development" column. The WIP limit for this column was first set to 3, thus allowing each developer to work on only one user story at any given moment. This WIP limit appeared to be too low due to substantial number of stories that were rejected by the Product Owner during "Acceptance". These stories were returned back to the "Next" column and treated as "silver bullets" having higher priority than other stories. In order to get these stories done as quickly as possible and shorten the lead time, the WIP limit of the "Development" column was increased to 6. However, we think that in normal circumstances an optimal value should be 4 or 5.

On the other hand, the WIP limit for "Acceptance" (which also considered the fact that there were 3 developers submitting their stories for approval) worked well.

The "Deploy" column was omitted since the project was experimental and did not have a real customer. Therefore, the user stories accepted by the Product Owner moved directly to the final "Done" column.

5 Conclusions

Scrum and Kanban are not mutually exclusive competing methodologies, but can complement each other to facilitate higher performance in software development teams. Scrum provides a framework for iterative and incremental development process, whereas Kanban ensures high visibility of the workflow and quick identification of possible bottlenecks, thus enabling continuous process improvement.

The aim of this paper was to shed some light on possibilities of combining both methods by providing a description of a generic Kanban board, ground rules for its use, and an example of its adaptation to the needs of a real-life project.

The proposed board includes a full range of activities (steps in the development process) and approaches to board modeling (working states, buffers) that should be taken into account when deciding about the most appropriate board structure. In case of complex projects, it can be used as such; however, it is primarily intended as a reference from which the users can derive the structure that best suits their development process.

We hope that our paper will contribute to better understanding of Kanban and help software engineering professionals who plan to introduce Scrumban in their development process to find an optimal way for implementation.

References:

[1] L. Williams, Agile Software Development Methodologies and Practices, *Advances in Computers*, Vol. 80, 2010, pp. 1–44.

[2] Manifesto for Agile Software Development, http://www.agilemanifesto.org (last accessed 7.11.2013).

[3] S. W. Ambler, Has Agile Peaked? Let's Look at the Numbers. *Dr. Dobb's Journal*, May 2008, http://www.ddj.com/architect/207600615?pgno=1 (last accessed 28.10.2013).

[4] M. Ceschi, A. Silitti, G. Succi, S. De Panfilis, Project Management in Plan-Based and Agile Companies, *IEEE Software*, Vol. 22, No. 3, 2005, pp. 21–27.

[5] C. Mann, F. Maurer, A Case Study on the Impact of Scrum on Overtime and Customer Satisfaction. In: *Proceedings of the Agile Development Conference (ADC'05)*, Denver, USA, 2005, pp. 70–79.

[6] K. Schwaber, J. Sutherland, *Software in 30 Days*, John Wiley and Sons, Hoboken, NJ, 2012.

[7] D. Cohen, M. Lindvall, P. Costa, An Introduction to Agile Methods, *Advances in Computers*, Vol. 62, 2004, pp. 2–67.

[8] D. West, T. Grant, Agile Development: Mainstream Adoption Has Changed Agility, Forrester, Januray 20, 2010.

[9] VersionOne, 7th Annual State of Agile Development Survey, http://www.versionone.com/state-of-agile-survey-results/ (last accessed 28.10.2013).

[10] L. Rising, N. S. Janoff, The Scrum Software Development Process for Small Teams, *IEEE Software*, Vol. 17, No. 4, 2000, pp. 26–32.

[11] K. Schwaber, *Agile Project Management with Scrum*, Microsoft Press, Redmond, WA, 2004.

[12] D. Anderson, *Kanban – Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, Sequim, WA, 2010.

[13] C. Ladas, *Scrumban – Essays on Kanban Systems for Lean Software Development*, Modus Cooperandi, Seattle, WA, 2008.

[14] T. Ohno, *Toyota Production System – Beyond Large-scale Production*, Productivity Press, Portland, OR, 1988.

[15] E. Corona, F. E. Pani, A Review of Lean-Kanban Approaches in the Software Development. *WSEAS Transactions on Information Science and Applications*, Vol. 10, No. 1, 2013, pp. 1–13.

[16] D. I. K. Sjøberg, A. Johnsen, J. Solberg, Quantifying the Effect of Using Kanban Versus

Scrum: A Case Study. *IEEE Software*, Vol. 29, No. 5, 2012, pp. 47–53.

[17] D. J. Anderson, J. Concas, M. I. Lunesu, M. Marchesi, H. Zhang, A Comparative Study of Scrum and Kanban Approaches on a Real Case Study Using Simulation. In C. Wohlin (Ed.): *XP 2012, Lecture Notes in Business Information Processing 111*, 2012, pp. 123–137.

[18] M. Ikonen, P. Kettunen, N. Oza, P. Abrahamsson, Exploring the Sources of Waste in Kanban Software Development Projects, In: *Proceedings of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2010, pp. 376–381.

[19] M. Ikonen, Leadership in Kanban Software Development Projects: A Quasi-controlled Experiment, In P. Abrahamsson and N. Oza (Eds.): *LESS 2010, Lecture Notes in Business Information Processing 65*, 2010, pp. 85–98.

[20] R. Polk, Agile and Kanban in Coordination, In: *Proceedings of 2011 Agile Conference*, pp. 263–268.

[21] N. Nikitina, M. Kajko-Mattsson, M. Stråle, From Scrum to Scrumban: A Case Study of a Process Transition, In: *Proceedings of International Conference on Software and Systems Process*, Zurich, Switzerland, June 2-3, 2012, pp. 140–149.

[22] H. Kniberg, M. Skarin, *Kanban and Scrum – Making the Most of Both*, C4Media Inc., 2010.

[23] J. Little, Why I Prefer ScrumBan to Kanban, http://www.leanagiletraining.com/blog/key-problems/why-i-prefer-scrumban/ (last accessed 7.11.2013).

[24] M. Cohn, *User Stories Applied for Agile Software Development*, Pearson Education, Boston, MA, 2004.

[25] M. Cohn, *Agile Estimating and Planning*, Prentice Hall PTR, Upper Saddle River, NJ, 2006.

[26] J. Grenning, Planning Poker or How to Avoid Analysis Paralysis While Release Planning, 2002, http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf (last accessed 28.10.2013).

[27] V. Mahnic, T. Hovelja, On Using Planning Poker for Estimating User Stories, *Journal of Systems and Software*, Vol. 85, No. 9, 2012, pp. 2086–2095.

[28] V. Mahnic, Applying Kanban Principles to Software Development, In: *Proceedings of the 16th International Conference on Information Technology for Practice*, Ostrava, Czech Republic, October 10-11, 2013, pp. 89-96.