Kanban in software engineering education: an outline of the literature

Viljan Mahnič

University of Ljubljana
Ljubljana, Slovenia

ABSTRACT: Kanban, an agile software development methodology, is growing increasingly popular in the software industry. In order to meet industry needs, universities must find a way to integrate Kanban into their software engineering courses. In this article is provided an outline of existing literature on this topic. The search of studies in the Scopus database revealed 13 papers that were identified as primary studies relevant to this research. These studies are classified by their main topic into four groups, viz. 1) teaching Kanban through practical work on student projects; 2) teaching Kanban through educational games; 3) reporting students' perceptions of Kanban and/or Scrumban (a related methodology) and 4) position papers. A short description of each study and its main results are provided. It was found that the literature on the use of Kanban in software engineering education is sparse and of limited quality. Only a few studies contain discussions on teaching the most important Kanban concepts (i.e. the structure of the Kanban board, setting of work-in-progress limits, maximising workflow and measuring lead time) in sufficient detail.

INTRODUCTION

Although Scrum is still the most widely used agile software development methodology, the use of Kanban is increasing steadily. The last three State of Agile reports by VersionOne indicate the increasing use of Kanban techniques, from 39% in 2015 to 50% in 2016, to 65% in 2017. Additionally, the Kanban board (a physical or digital project management tool) was reported the most used project management tool in 2017. Teaching Kanban thus is an important move in the education of future software engineering professionals. In order to meet industry needs, universities must integrate Kanban into their software engineering courses.

Unfortunately, the literature on the use of Kanban in software engineering is very sparse. An early review identified only 19 primary studies, but none of them addressed educational issues [1]. A recent systematic mapping study of the state-of-the-art of Kanban research again confirmed the lack of rigorous studies. Only 23 primary studies were found; three of them in the area of software engineering education [2]. However, these studies were only analysed from the research perspective and not from the pedagogical one.

The aim of this paper is to focus on educational issues and provide a review of the literature dealing with Kanban in software engineering courses. The review aims to present 1) what has been reported about the use of Kanban in software engineering education and 2) the typical approaches most frequently used in teaching practice. Special attention is paid to teaching the principles of the Kanban board organisation, setting work-in-progress (WIP) limits, optimising workflow, and measuring the lead time, which seem to be the most important issues when introducing Kanban to an industrial environment [3]. Since Kanban can be used as a standalone method or in combination with Scrum, studies on Scrumban are also included.

In order to answer these questions, a review protocol was developed following the guidelines suggested by Kitchenham and Charters that enabled the selection of the most important primary studies [4]. These studies were then classified by their topics and further analysed with regard to the approach to teaching Kanban. The next section describes the selection of primary studies and their classification into four groups. Then, each group is described in a separate section. Finally, the most important conclusions are provided.

IDENTIFICATION AND SELECTION OF STUDIES

Relevant studies were searched for in the Scopus electronic database in December 2018 using the search string (TITLE-ABS-KEY(Kanban OR Scrumban) AND TITLE-ABS-KEY(software engineering) AND TITLE-ABS-KEY(course OR education OR teaching)), which means that all publications containing the words *Kanban* or *Scrumban*, *software engineering* and *course*, *education* or *teaching* in their title, abstract or keyword list were expected to be retrieved.

The substring *Kanban OR Scrumban* was used to broaden the search scope by also including studies dealing with teaching Kanban in combination with Scrum. The substring *software engineering* was used to eliminate studies dealing with the use of Kanban in other areas (e.g. manufacturing). The purpose of the substring *course OR education OR teaching* was to cover the widest range of studies in the field of education. The Scopus database was chosen, because it provides a single source of a wide range of high quality publications that would otherwise have to be searched for in different digital libraries (e.g. ACM Digital Library, IEEE Explore, ISI Web of Science).

The search resulted in 16 matches, out of which three were excluded after reading the abstract for being a preface to conference proceedings, describing improvement of software development in a company and dealing with manufacturing management, respectively. The remaining studies were read in full. Unfortunately, it was found that most of them lacked a precise description of how the most important Kanban concepts are taught and how the corresponding courses are organised. Some of them provide only survey results, a discussion of general guidelines on how to teach agile methods or just mention the use of some Kanban elements (e.g. Kanban board) in the context of a wider student project.

Nevertheless, it was decided to include all the remaining studies in the review, thus providing the readers with as complete a picture as possible of issues concerning the use of Kanban in software engineering education. The 13 primary studies chosen for review are listed in alphabetic order of the first author's name at the end of this article, as references [5-17]. For further discussion, the studies are classified by their main topic into four groups, as shown in Table 1.

Table 1: Classification of studies by topic.

Group	Topic	Studies dealing with this topic
1	Teaching Kanban through practical work on student projects	10, 11, 12, 13, 17
2	Teaching Kanban through educational games	7, 16
3	Students' perceptions of Kanban and/or Scrumban	5, 6, 14,
4	Position papers	8, 9, 15

TEACHING KANBAN THROUGH PRACTICAL WORK ON STUDENT PROJECTS

Studies from this group provide a description of the design of courses that require students to work in teams in order to develop a non-trivial software project. A study by Mahnič describes a capstone course consisting of three weeks of formal lectures and 12 weeks of practical work [10]. During the first three weeks, the students attend formal lectures on agile software development methods (in particular Scrum and Kanban) and get acquainted with the product backlog consisting of a set of user stories they are going to develop. The product backlog is prepared by a domain expert playing the role of the product owner.

To develop the required functionality, the students work in teams of four that are divided into two groups: the Scrumban group and the Kanban group. Teams belonging to the Scrumban group retain the Scrum concept of fixed-length iterations. Consequently, their work is divided into three regular Scrum sprints, which are augmented by the use of the Kanban board and WIP (work in progress) limits. The Kanban board includes the sprint backlog column, which is initiated at each sprint planning meeting. The content of the sprint backlog must not exceed the estimated team velocity.

Teams belonging to the Kanban group follow lean concepts more strictly by abandoning fixed-length iterations and sprint planning. Instead, the product owner maintains a small number of high priority stories, which a team member can pull into development whenever he/she completes the user story he/she worked on before. In order to ensure continuous workflow, the product owner is also expected to evaluate user stories promptly, as soon as each user story is signalled as finished. Consequently, the review meetings are not held at regular intervals, but are event-driven. A review meeting is triggered whenever there is a set of minimum marketable features, defined by the product owner, ready for release.

The study provides examples of Kanban boards used, discusses adaptation of WIP limits, and includes empirical data on the average lead time reported by student teams at retrospective meetings. The initial Kanban board structure is prescribed by instructors, but later can be adapted to better reflect the operation of each team. The WIP limits are introduced stepwise by gradually decreasing the number of user stories in development until a bottleneck occurs. By monitoring the average lead time the students are encouraged to search for improvements in their development process. Results of a survey among students are provided indicating that the students overwhelmingly are satisfied with the course and successfully grasped the main characteristics of Scrumban and Kanban.

Another study by Mahnič [11] can be considered as a supplement to the aforementioned study [10] providing a more detailed description of tool support for execution of student projects. In the second part, it is shown how Kanbanize (www.kanbanize.com), a commercial project management tool, is used for managing Kanban-based projects. The tool makes it possible to create a Kanban board with an arbitrary structure, thus allowing students to specify the columns that best suit their development process. Each column can be further divided into sub-columns and sub-sub-columns. Each column can have a WIP limit indicating how many cards can be in the corresponding workflow state at any one time.

By using the Kanbanize Analytics, the student teams can obtain a cumulative flow diagram, compute the lead time, and compare the time that a card stayed in each column to the amount of work actually spent in that column. These data have to be reported at each retrospective meeting and serve as measures of process effectiveness. Examples of the Kanban board, the cumulative flow diagram and the lead time calculation from an actual student project executed in the academic year 2017/18 are provided.

A study by Matthies provides a description of a capstone course design, which adds hands-on learning of the lean principles advocated by Kanban into a capstone project run with Scrum [12]. The course lasts 15 weeks and combines both traditional lectures, introducing students to agile methods, and practical project work, requiring students to develop a web-application using the Ruby on Rails framework. In order to allow students to get started with hands-on project work as fast as possible, two practical exercises are employed at the beginning of the course: A LEGO Scrum exercise and an automated programming tutorial.

Practical project work is divided into four Scrum sprints, followed by a so-called Kanban week. Each sprint lasts two weeks and includes sprint planning, sprint review, sprint retrospective and weekly stand-up meetings. Kanban is introduced in a lecture towards the end of the course, after the students have already gained experience with Scrum. During the lecture, Kanban boards and the concepts of visualising workflow and limiting WIP are presented in detail. These concepts are then tried out in students' projects during the last week-to-10 days of the project instead of the last Scrum sprint. Unfortunately, the study does not provide concrete examples of Kanban boards and WIP limits actually used.

On the other hand, the study provides an evaluation of the course consisting of a survey at the very end of the course, and a comparative analysis of the development artefacts produced during Kanban and Scrum project work. The survey revealed positive student attitudes towards the course, as well as to the introduction of Kanban at the end of the project, when the nature of work changes from new feature implementation to bug fixing and code polishing.

A study by Neyem et al refers to the software engineering capstone course at the Computer Science Department of the Engineering School at Pontificia Universidad Católica de Chile [13]. The study does not provide a description of the content of the course and teaching methods, but does have an explanation of the design and functionality of a software tool that supports the management of students' projects. The tool involves a series of features that revolve around the Kanban approach (e.g. the Kanban board and the cumulative flow diagram) and offer better visualisation and monitoring of the activities defined in a project during different stages.

A study by Soares and Ferrão also does not focus on Kanban, but on continuous integration and project-based learning strategies that allow a large group of students to plan their project together and keep team members aware of project progress at any time [17]. Scrum practices are used to manage the project, while students share an online Kanban board that supports them in organising their sprints. A screenshot of the board is provided but without an explanation of its structure and the use of WIP limits.

TEACHING KANBAN THROUGH EDUCATIONAL GAMES

The use of games is described in the studies by Heikkilä et al [7] and Scharlau [16]. The first study provides an in-depth analysis of the use of the GetKanban collaborative board game (http://getkanban.com) [7], while the second study presents an overview of games that can be used to familiarise students with agile approaches before they start working on a group project [16].

The authors of the first study slightly modified the GetKanban game in order to teach Kanban and lean thinking in a Master's degree software project management course at Aalto University, Finland. The game is played by a team of players whose overall goal is to produce financial value, which is mainly done by gaining new subscribers which, in turn, is achieved by producing new features. In order to produce features most efficiently, the players must decide how to assign workers to the features and control WIP [7].

The game uses three types of tickets: *normal* tickets (representing new features); *fixed delivery date* tickets (creating benefit or loss if (not) completed in time); and *expedite* tickets (having special rules). The board consists of six columns: *Ready*; *Analysis*; *Development*; *Test*; *Ready to Deploy*; and *Deployed*. The *Analysis* and *Development* columns are further split into *In Progress* and *Done* sub-columns. The WIP limits are set in *Ready*, *Analysis*, *Development* and *Test* columns in order to maximise workflow. A separate *Expedite* lane with WIP limit of one is introduced to contain the *expedite* tickets. The game is played in rounds, each round representing a working day. At the end of each round, the daily event cards are used to introduce additional game rules that allow students to experience the effect of different decisions on the game's outcome.

A special feature of the study by Heikkilä et al is a comprehensive evaluation of the learning outcomes regarding eight learning goals [7]. The students were asked to answer three questionnaires. The *pre-course* questionnaire was filled before the first lecture of the course for the purpose of self-evaluation of the subjects taught during the course. The *pre-game* and *post-game* questionnaires were filled before and after the GetKanban game. Both questionnaires comprised Kanban- and Lean-related true/false statements and questions for the self-evaluation of knowledge on Kanban.

Additionally, the *post-game* questionnaire contained some statements to evaluate the GetKanban game. For the purpose of qualitative analysis, the students were also required to submit a learning diary describing reflections and giving feedback on the game.

The analysis of differences between the *pre-* and *post-game* questionnaires has shown a statistically significant improvement in students' opinions on their knowledge. However, the analysis of actual learning by comparing students' answers on true/false statements before and after the game revealed that the answers did not change in a statistically significant way. Although the students perceived they had learned substantially from the game and evaluated the game very positively, the study by Heikkilä et al concludes that the learning goals were only partially reached and that educational games may not be more effective in teaching software engineering than more traditional methods [7].

A study by Scharlau provides a short description of six games that the author uses with Master's degree information technology students in order to provide insight into the thinking and rationale of agile, lean and Kanban approaches [16]. One of the games is the Kanban game that enables students to experience what happens to the workflow when limits are placed on the number of items that can be worked on at each stage of the project. The game is played in three iterations using different types of Kanban boards during each round. At the beginning, the board consists of three columns: *to do*; *in progress;* and *done*. In the second round, the *in progress* column is broken down into *design*, *develop* and *test*. Finally, queues and WIP limits are added.

The students play the games with the same teams used later for a 12-week summer project with a real client. The author claims that the use of games diminishes the need for formal lectures, fosters team building, increases students' confidence in their abilities, and increases their awareness of the need for regular communication with the clients. After replacing formal lectures with games, the students became more engaged and the quality of their projects improved.

STUDENTS' PERCEPTIONS OF KANBAN AND/OR SCRUMBAN

Studies from this group provide a description of the results of different surveys among students performed after completion of practical project-oriented courses that exposed students to some Kanban or Scrumban practices. However, they do not provide a detailed description of how Kanban/Scrumban was taught, but concentrate on other issues. A study by Ahmad et al describes students' opinions on the *software factory*, a learning environment built at the University of Oulu, Finland, to provide a realistic environment for project-based courses [5]. A subsequent study by Ahmad et al [6] details the earlier results that refer to the use of Kanban [5]. A study by Plengvittaya and Sanpote [14] analyses students' opinions on Scrumban in view of critical success factors in agile software projects reported in the literature [18].

The software factory described in the study by Ahmad et al consists of one big and three small rooms equipped with computers, software development tools and Kanban boards [5]. An example of a Kanban board is provided, but it only serves as an illustration of a Kanban board and does not belong to an actual project executed in the software factory. Considerations regarding the choice of board columns and WIP limits are also not discussed. Instead, the study concentrates on students' perceptions and attitudes towards the software factory as a whole.

For this purpose, two previously defined instruments by Newby and Fisher (i.e. computer laboratory environment inventory, and attitude towards computers and computing courses questionnaire [19]) were augmented to include two more constructs: the Kanban board and collaborative learning. The survey was answered by 19 (out of 28) students of the Master's degree programme, who took part in seven projects in 2013. The results showed that overall the perception about the Kanban board was positive (an average mean of 3.59 on a five-point Likert scale).

The mentioned study by Ahmad et al [6] describes the same learning environment, but provides a more detailed analysis of questions regarding student opinions on Kanban and competencies gained. It is based on answers from 51 (out of 96) Master's degree students who participated in 96 software factory projects in the 2013 academic year and spring 2014 semester. Results show that Kanban helps students understand the essential aspects of a software project by helping them to identify bottlenecks, to understand better project activities and to obtain a better project overview. The great majority of students expressed their willingness to use Kanban in real software projects. They also felt that teaching software engineering in the software factory using Kanban provided them with relevant skills and competencies needed for professional work in the software industry.

The last study in this group provides a description of the results of a survey among third-year undergraduate students of the University of Phayao, Thailand. They attended the Software Project Management course in the first semester of 2017 [14]. There were 18 students divided into six teams who used Scrumban for managing their projects. The survey was based on previous research that classified critical success factors in agile software projects into five dimensions: organisational; people; process; technical; and project [18]. For each dimension (except organisational, which the authors considered not relevant for the study), the students were provided with a set of statements about how Scrumban contributes to project success. Students' agreement with these statements was measured using a five-point Likert scale. The great majority of students agreed with all suggested statements (the mean values were between 3.89 and 4.56), thus indicating that the use of Scrumban has a positive impact on the project success.

POSITION PAPERS

Position papers suggest different approaches on how to teach agile methods and software project management. The two studies by Kropp and Meier are very similar [8][9]. Both propose a more holistic approach for teaching agile software engineering, in which the required agile practices and values are not only integrated theoretically but also practically applied and repeated until they become a habit to students. They introduce the so-called *pyramid of agile competences* consisting of three levels: engineering practices; management practices; and agile values. Engineering practices can be taught in the classroom, management practices are best taught through teamwork projects, while agile values are the most difficult to teach, since they often require a change in the attitude of an individual.

The pyramid is also a visualisation of the order in which the different practices should be taught. Good knowledge of engineering practices is a prerequisite for teaching management practices, which (together with engineering practices) represent a foundation for understanding agile values. Such order should be taken into account both in the design of the content of individual software engineering modules and in the design of the curriculum as a whole.

As an example of the integrated teaching approach, a description is provided of a 16-week Software Engineering course at the ZHAW Zurich University of Applied Sciences. In the first part of the course, the engineering practices of Extreme Programming are introduced, while the second part concentrates on practical project work using the management practices of Scrum. The agile values are imparted to students by propagating them as working values throughout the course.

The second study by Kropp and Meier also provides an example of how the proposed approach was implemented at the University of Applied Sciences and Arts Northwestern Switzerland within the scope of two consecutive courses [9]. The Software Construction course is a focus on the teaching of agile engineering skills, while the follow-up Software Project Management course is a concentration on management practices.

A study by Ralph appeared as one of the search results because *Kanban* is one its keywords [15]. However, in its text the word *Kanban* does not appear at all and the study should have been excluded from the review as irrelevant. Nevertheless, it was decided to mention it as a position paper, because it gives a different perspective on teaching software project management, as compared to others. Its author advocates the thesis that software project management curricula should be grounded in relevant software engineering and management theory, rather than specific software development methods, process models, project management frameworks and other practitioner-generated prescriptions. A course design is presented as an alternative to the knowledge areas in the ACM/IEEE model curriculum.

CONCLUSIONS

Confirmed in this review are the findings of other studies by Ahmad et al that the literature on the use of Kanban in software engineering is very sparse and of limited quality [1][2]. As shown in Table 2, the first studies did not appear until 2013, but the number of studies increased significantly in 2018, almost reaching the number of studies in all previous years. This indicates that the universities became aware of the importance of teaching Kanban and started following the trends in the industry.

Table 2: Classification of studies by year of publication.

Year	2013	2014	2015	2016	2017	2018
Number of studies	2	3	1	1	0	6

The review revealed that two types of teaching Kanban prevail: learning through practical project work and learning through educational games. The great majority of studies report that the students are overwhelmingly satisfied with these approaches. The results are similar to those of the study on teaching Scrum and indicate that agile methods are best taught through practical problem solving [20].

Of all the Kanban concepts, most of the studies mention the use of Kanban board. The Kanban board is also used in studies where the primary goal is not teaching Kanban, but the board serves as a means for better monitoring of students' projects. Unfortunately, only a few studies detail the structure of the board and the implementation of other important Kanban concepts, such as limiting work in progress, maximising workflow and measuring lead time. Evaluation of the approaches used is also often inadequate. In this respect, the studies by Heikkilä [7] and Mahnič [10] positively stand out from the others. The study by Matthies also provides a good description of the course design, but is more Scrum-oriented, thus lacking details regarding Kanban concepts [12].

Teaching the combination of Scrum and Kanban is described in studies by Mahnič [10] and by Matthies [12]. In the former study, Scrum and Kanban are combined into a single method (i.e. Scrumban) with the aim of making the most of both [10]. In the later study, the methods are taught in a sequence: the first four sprints are run according to Scrum, while Kanban is used as a replacement of the fifth sprint [12]. The use of Scrumban is also mentioned in the study by

Plengvittaya and Sanpote, but without details on how its concepts are imparted to students [14]. In the future, the number of studies on teaching Kanban in the area of software engineering should increase and their quality should improve.

REFERENCES

1. Ahmad, M.O., Markkula, J. and Oivo, M., Kanban in software development: a systematic literature review. *Proc. 39th Euromicro Conf. on Software Engng. and Advanced Applications*, Santander, Spain, 9-16 (2013).
2. Ahmad, M.O., Dennehy, D., Conboy, K. and Oivo, M., Kanban in software engineering: a systematic mapping study. *J. of Systems and Software*, 137, 96-113 (2018).
3. Kniberg, H. and Skarin, M., *Kanban and Scrum - Making the Most of Both*. C4Media Inc. (2010).
4. Kitchenham, B. and Charters, S., Guidelines for performing systematic literature reviews in software engineering (2007), 3 December 2018, http://www.elsevier.com/__data/promis_misc/525444systematicreviewsguide.pdf
5. Ahmad, M.O., Liukkunen, K. and Markkula, J., Student perceptions and attitudes towards the software factory as a learning environment. *IEEE Global Engng. Educ. Conf.*, Istanbul, Turkey, 422-428 (2014).
6. Ahmad, M.O., Markkula, J. and Oivo, M., Kanban for software engineering teaching in a software factory learning environment. *World Trans. on Engng. and Technol. Educ.*, 12, **3**, 338-343 (2014).
7. Heikkilä, V.T., Paasivaara, M. and Lassenius, C., Teaching university students Kanban with a collaborative board game. *Proc. 38th IEEE Inter. Conf. on Software Engng. Companion*, Austin, Texas, USA, 471-480 (2016).
8. Kropp, M. and Meier, A., Teaching agile software development at university level: values, management, and craftsmanship. *Proc. 26th Conf. on Software Engng. Educ. and Training*, San Francisco, USA, 179-188 (2013).
9. Kropp, M. and Meier, A., New sustainable teaching approaches in software engineering education: how agile methods influence teaching software engineering. *IEEE Global Engng. Educ. Conf.*, Istanbul, Turkey, 1019-1022 (2014).
10. Mahnič, V., From Scrum to Kanban: introducing lean principles to a software engineering capstone course. *Inter. J. of Engng. Educ.*, 31, **4**, 1106-1116 (2015).
11. Mahnič, V., Tools support for the management of students' software engineering projects. *World Trans. on Engng. and Technol. Educ.*, 16, **3**, 218-223 (2018).
12. Matthies, C., Scrum2Kanban: integrating Kanban and Scrum in a software engineering capstone course. *Proc. ACM/IEEE Inter. Workshop on Software Engng. Educ. for Millennials*, Gothenburg, Sweden, 48-55 (2018).
13. Neyem, A., Diaz-Mosquera, J. and Benedetto, J.I., A cloud-based mobile system to improve project management skills in software engineering capstone courses. *Mobile Infor. Systems*, article ID 6371793, 16 (2018).
14. Plengvittaya, C. and Sanpote, D., Scrumban for teaching at undergraduate program: a case study from software engineering students. *Proc. 3rd Inter. Conf. on Digital Arts, Media and Technol.*, Chiangray, Thailand, 109-114 (2018).
15. Ralph, P., Re-imaging a course in software project management. *Proc. 40th Inter. Conf. on Software Engng.: Software Engng. Educ. and Training*, Gothenburg, Sweden, 116-125 (2018).
16. Scharlau, B., Games for teaching software development. *Proc. 18th Annual Conf. on Innov. and Technol. in Computer Science Educ.*, Canterbury, UK, 303-308 (2013).
17. Soares, L.P. and Ferrão, R., Continuous integrating team learning. *Proc. Inter. Symp. on Project Approaches in Engng. Educ.*, Brasilia, Brasil, 202-209 (2018).
18. Chow, T. and Cao, D-B., A survey study of critical success factors in agile software projects. *J. of Systems and Software*, 81, **6**, 961-971 (2008).
19. Newby, M. and Fisher, D., An instrument for assessing the learning environment of a computer laboratory. *J. of Educational Computing Research*, 16, **2**, 179-190 (1997).
20. Mahnič, V., Scrum in software engineering courses: an outline of the literature. *Global J. of Engng. Educ.*, 17, **2**, 77-83 (2015).