Tools support for the management of students' software engineering projects

Viljan Mahnič

University of Ljubljana
Ljubljana, Slovenia

ABSTRACT: Two software tools that are used at the University of Ljubljana, Ljubljana, Slovenia, to support the management of students' software engineering capstone projects are described. The first tool is an in-house developed tool that supports the management of Scrum-based projects taken within the capstone course at the first Bologna level. The tool not only provides the typical functionality required for managing Scrum projects, but also meets the specific needs of teaching staff and researchers. The second tool is Kanbanize, a commercially available tool used for managing students' projects at the second Bologna level. This tool supports the most important Kanban concepts, i.e. the maintenance of the Kanban board, limiting work in progress and optimising the lead time.

INTRODUCTION

Learning software engineering (SE) requires a lot of practical work that helps students to fully understand and reflect on SE theory. Therefore, it is suggested that the SE curriculum should include project-based classes that provide an adequate combination of theory and practice [1]. Recommendations for undergraduate software engineering curricula also mandate that students undertake a capstone course, which covers one full year [2]. The course should integrate previously learned material, deepen their understanding of that material, extend their area of knowledge, and apply their knowledge and skills in a realistic simulation of professional experience.

At the University of Ljubljana, a SE capstone course is taught on two levels: the first and the second Bologna level. At both levels, the course lasts one semester (15 weeks). At the first Bologna level, the course is mandatory for all students in the SE curriculum, while at the second Bologna level, it has been declared as an optional professional course. In addition to these basic goals that all capstone courses should pursue, the capstone courses at the University of Ljubljana also serve as a tool for imparting agile and lean methods to students [3]. Projects that students develop at the first Bologna level strictly follow Scrum rules and practices, while projects at the second Bologna level are used to introduce lean concepts of Kanban.

Teaching agile and lean software development through capstone projects not only exposes students to state-of-the-art topics having industrial relevance, but also stimulates the teaching staff to use modern ways of teaching through project-based learning, and makes it possible to conduct empirical studies with students, thus contributing to empirical evidence regarding agile and lean practices. Since students work in groups on quasi-real projects, monitoring their progress and collecting data for empirical studies is best done by the use of a computerised tool. The tool should support the development methodology (i.e. Scrum and Kanban, respectively), help teaching staff to monitor students' progress and reduce the burden of administrative work and provide empirical data for evidence-driven assessment of the development process.

The purpose of this article is to describe two tools that are used for the management of students' capstone projects at the University of Ljubljana. The first one is an in-house developed tool that supports the management of Scrum-based projects taken within the capstone course at the first Bologna level. The second one is Kanbanize, a commercially available tool used for managing students' projects at the second Bologna level.

The remainder of the article is structured as follows: the main characteristics of Scrum-based capstone projects and the corresponding tool are provided in the next two sections, followed by two sections describing the Kanban projects and the Kanbanize tool, respectively. The last section provides a conclusion.

SCRUM-BASED CAPSTONE PROJECTS

Scrum prescribes iterative and incremental development processes. All work is done in iterations (called sprints) and each sprint should provide an increment of potentially shippable functionality. There are three roles: the product owner, the team and the Scrum master. The product owner represents interests of everyone with a stake in the project and its results. He/she maintains a prioritised list of requirements (called product backlog), which usually consists of a set of user stories. The team is responsible for implementation of the required functionality, while the Scrum master is responsible for ensuring that everybody follows Scrum rules and practices.

The Scrum-based capstone projects comprise four sprints [4]. The first sprint (also called sprint 0) lasts three weeks and serves as a preparatory sprint. During sprint 0, intensive formal lectures take place in order to teach students Scrum and acquaint them with user stories they are going to develop. The stories are written and prioritised by a domain expert (the teacher or a representative of a co-operating company) playing the role of the product owner. Students are grouped into teams of four, responsible for the development of the required functionality. Each team estimates the stories using *planning poker* or *team estimation game* and prepares the release plan.

The rest of the course is divided into three regular Scrum sprints. Each sprint starts with a sprint planning meeting at which student teams define the contents of the next iteration and develop the initial version of the sprint backlog. During the sprint, the teams have to meet regularly at the daily Scrum meetings and maintain their sprint backlogs, while at the end of each sprint, the sprint review and sprint retrospective meetings take place. At the review, the students present their results to the instructors, while at the retrospective meeting students and instructors meet to review the work done in the previous sprint, giving suggestions for improvements in the next. After three sprints, the first release should be complete and delivered to the customer.

Students are required to provide data on their initial effort estimates, the amount of work spent and the amount of work remaining. At the beginning of each sprint, they are encouraged to re-estimate their velocity and the remaining user stories in order to obtain a more realistic plan for future iterations. Instructors compare students' plans with actual achievements and analyse whether students' estimation and planning abilities improve as they gain more knowledge of Scrum and a better understanding of the user requirements. Special attention is devoted to the notion of *done* requiring the students to bring the code up to the useful, real-world level, which can survive an encounter with end users.

Data collected during the course not only provide feedback about what the students have learned, but also contribute to evidence-driven assessment of Scrum processes and practices. Students' perceptions of Scrum were studied first [5]; afterwards, the course served as a case study on agile estimating and planning using Scrum [6], and provided data for the evaluation of the accuracy of *planning poker* estimates [7].

TOOL SUPPORT FOR SCRUM-BASED CAPSTONE PROJECTS

Motivation for developing a tool for the management of Scrum-based capstone projects stemmed from results of the survey [8], which found that existing agile tools are either too simple, providing only limited functionality or too cumbersome and difficult to use. Additionally, the survey found existing tools difficult to adapt to the specific needs of end users.

Sprint 0

During sprint 0, the tool: 1) automates the formation of student teams and the allocation of teams to appropriate sections of laboratory classes; 2) provides facilities for initial product backlog preparation; and 3) automates the effort estimation process. Considering the Scrum principle of self-organisation, students are given an opportunity to decide who they should work with. When a team is formed, the students use the tool to enter the team composition and choose the section of laboratory classes they want to attend, as well as the project they are going to develop. The choice of the project and the desired laboratory class section is performed on the basis of the first-come, first-served principle.

In parallel with the student team formation, the member of the teaching staff (or the representative of a company) that plays the role of product owner uses the tool to prepare the initial product backlog, which consists of a set of user stories. As shown in the upper part of Figure 1, each user story is represented with a story card containing a written description (used for planning and as a reminder for further conversations) and a set of acceptance tests (to determine when a story is *done*). The product owner also defines the priority and business value.

At the end of sprint 0, the tool enables student teams to estimate the effort required for implementation of each user story. The tool automates two estimation techniques, i.e. *planning poker* and the *team estimation game*. Both techniques require user stories to be estimated in story points, and the tool allows each team member to provide his/her estimate either by using predefined values (i.e. 0.5, 1, 2, 3, 5, 8, 13, 20, 40) or by specifying an arbitrary explicit value.

Sprints 1, 2 and 3

Each regular sprint starts with the sprint planning meeting at which the user stories that a team commits to implement are marked by ticking the corresponding check box and moved to the sprint backlog with a single click. Then, the stories in the sprint backlog are decomposed into tasks and the effort required for each task is estimated. Finally, the tasks are assigned to team members. For each task, the tool enables recording of the effort spent and the amount of work remaining. An example of decomposition of a user story into tasks is shown in the lower part of the Figure 1 together with the status of each task, the assignee and the amount of work remaining.

#30 Kartotečni list

Priority: Must have | Business value: 0 | Actions: NOTES, EDIT TASKS

Sprint 3 Estimate: 4 pt Work: 7 h

Past Spr. Estimate: - Work: -

Študent/referentka/učitelj lahko izpiše kartotečni list.

Študent lahko izpiše kartotečni list samo zase, referentka in učitelj pa lahko izpišeta kartotečni list za kateregakoli študenta. Predmeti naj bodo grupirani po študijskih programih in študijskih letih. Za vsako študijsko leto se izpišejo vsi predmeti, ki jih je študent vpisal, ne glede na to, ali je izpit že opravil ali ne. Izpis naj bo možen v 2 variantah: vsa polaganja in samo zadnje polaganje.

Preveri izpis kartotečnega lista takoj po vpisu. Izpis mora obsegati vse predmete, ki jih je študent vpisal, brez ocen.
Vnesi nekaj ocen in preveri izpis na oba načina (vsa polaganja, samo zadnje polaganje).
Preveri za študenta, ki je vpisan v več študijskih letih.
Preveri za študenta, ki je vpisan v več študijskih programov.
Preveri izpis glave (študijsko leto, študijski program, letnik, vrsta vpisa, način študija) in noge (za vsako študijsko leto posebej število doseženih kreditnih točk in povprečna ocena).
Preveri izpis podatkov o vsakem polaganju (šifra predmeta, ime predmeta, izpraševalci, datum izpita, zaporedna številka polaganja v celoti, zaporedna številka polaganja v tekočem študijskem letu, število kreditnih točk, ocena).
Preveri izvoz v PDF in CSV.

Tasks	Status / action	Member	Remaining
Frontend	Completed	Blatnik B.	0 h
Backend	Assigned	Blatnik B.	1 h
Testiranje	Assigned	Blatnik B.	4 h
Generiranje pdf in csv	Pending	Černelč	3 h
		SUM:	8 h

Figure 1: An example of a user story card. The upper part of the figure represents the user story as it is described in the product backlog. The lower part shows the tasks that are added by the team during the sprint planning meeting.

Data on work remaining allow for the monitoring work progress through burn down charts, while the amount of work spent makes it possible to track the contribution of each student, and study the difference between the estimated and actual effort. The tool provides stacked burn down charts at two levels: sprint and release. Both charts visualise the correlation between the amount of work remaining and the progress of a student team in reducing this work. An example of a sprint burndown chart is given in Figure 2.

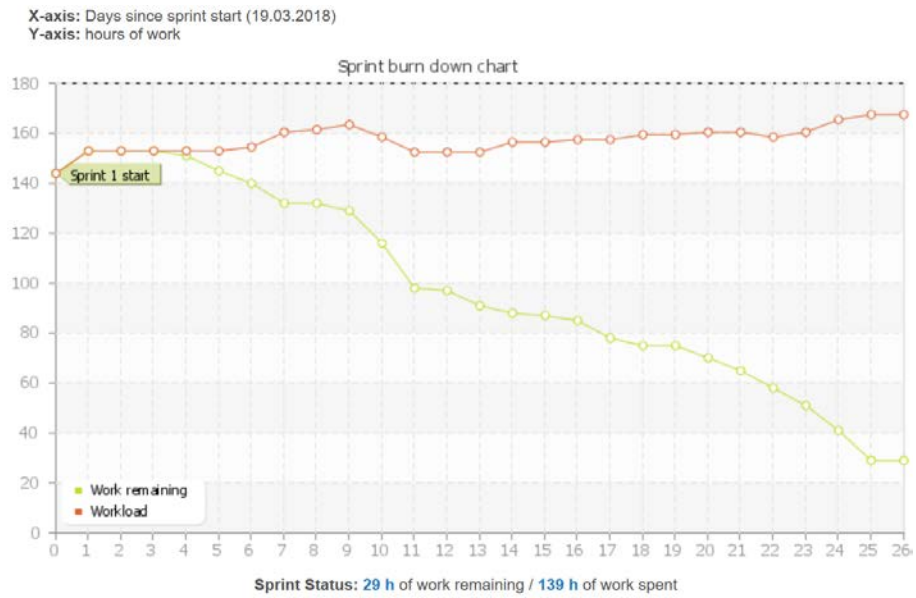

Figure 2: Sprint burndown chart.

For smooth running of the course, it is important that student teams work on their projects without procrastinating and the workload is distributed uniformly among team members. Figure 3 shows the so-called *team involvement chart*, which helps the students and the instructors to monitor these two aspects. The stacked bar chart in the upper part of the figure shows the amount of work performed by each student on each day of the sprint. In the lower part, the workload distribution is presented both in tabular form and with a pie chart.

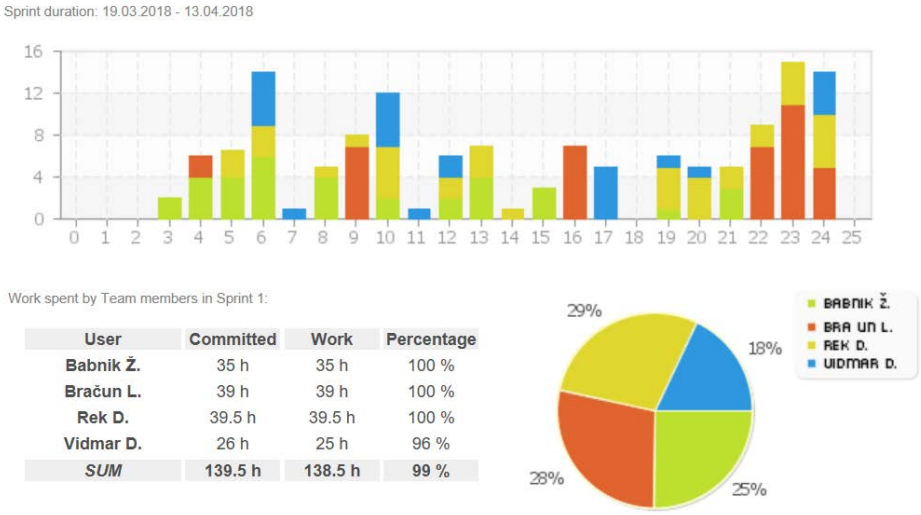

Figure 3: Team involvement chart. The upper part represents the amount of work spent by each team member on each day of the sprint. In the lower part, the contribution of each member is shown in tabular and graphical form.

The tool also provides information on whether all teams regularly hold daily Scrum meetings, and maintain data on work spent and work remaining. Students who fail to provide minutes of daily Scrum meetings or data on work spent and work remaining are automatically reminded via e-mail.

At the end of the sprint, the product owner marks the user stories that meet acceptance criteria as *done*, while unfinished stories are automatically moved back to the product backlog for possible inclusion in one of the subsequent sprints. For each rejected story, a comment can be added specifying the reasons for rejection.

End of the Project

At the end of the project, the tool assists the evaluation of students' projects by preparing a realisation report. On the basis of data collected during the project, the tool automatically computes the number of story points achieved by each student separately and the team as a whole. If there are several students working on the same user story, the story points are divided among the students relative to the effort they invested in the implementation.

KANBAN-BASED CAPSTONE PROJECTS

While retaining main characteristics of Scrum-based capstone projects, Kanban-based capstone projects also introduce the lean concepts of Kanban [9]. Special attention is devoted to issues, which seem to be most important, when introducing Kanban to software development, i.e. the structure of the Kanban board, assignment of work-in-progress (WIP) limits and measuring lead time. Additionally, different ways of introducing Kanban are studied: a stepwise introduction through Scrumban or *pure* Kanban approach abandoning some of the Scrum practices. For this reason, student teams are divided into two groups: the Scrumban group and the Kanban group.

Teams belonging to the Scrumban group (in the remainder Scrumban teams) retain the Scrum concept of fixed-length iterations. Consequently, their projects still consist of sprint 0 and three regular Scrum sprints, but are augmented by the use of the Kanban board and WIP limits. The board visualises the workflow, while the WIP limits prevent the team members from working on several work items at the same time, thus minimising lead time. The Kanban board includes the sprint backlog column, which is initiated at each sprint planning meeting. The contents of the sprint backlog must not exceed the estimated velocity.

Teams belonging to the Kanban group (in the remainder Kanban teams) follow lean concepts more strictly by abandoning fixed-length iterations and sprint planning. They are no longer required to maintain the sprint backlog and track their velocity. 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 a set of minimum marketable features (MMF) predefined by the product owner is ready for release.

There is no difference between the groups regarding daily Scrum and sprint retrospective meetings. Teams belonging to the Kanban group hold their retrospective meetings regularly at the same intervals as Scrumban teams (i.e. at the end of each Scrumban sprint) and all teams are required to meet regularly at the daily Scrum meetings.

TOOL SUPPORT FOR KANBAN-BASED CAPSTONE PROJECTS

The course execution is supported by Kanbanize, a commercial project management tool that has been developed especially for managing Kanban-based projects. The use of Kanbanize makes it possible to emphasise the most important Kanban concepts: process visualisation, limiting work in progress and measuring lead time.

In order to visualise the workflow, each user story from the product backlog is represented as a card on the Kanban board consisting of a sequence of columns that represent the various states a work item can exist in during the development process. Each student team maintains its own board and, as work progresses through the development lifecycle, moves the cards from one state to the other until they finish in the last column. By using Kanbanize, it is possible to create a Kanban board with an arbitrary structure. Users can thus define the columns that best suit their development process, and each column can be further divided into sub-columns and sub-sub-columns. Each column can have a WIP limit at the top, indicating how many cards can be in the corresponding workflow state at any one time.

Figure 4 shows an example of such a board. Note that due to limited space some columns are minimised (a special feature of Kanbanize), but can be expanded to look the same as those shown in their entirety.

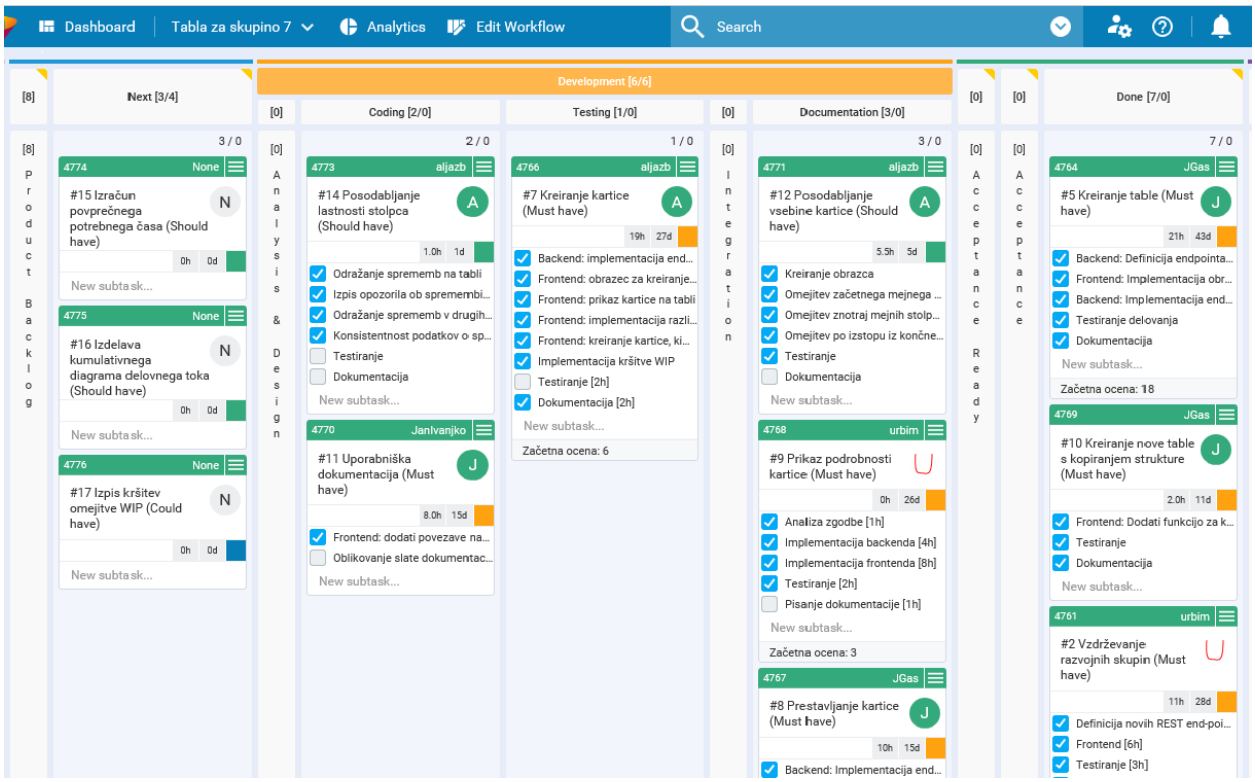

Figure 4: Kanban board used by a team belonging to the Kanban group.

When a new user story is created, the corresponding card is placed in the *product backlog* column. The *next* column is intended to contain a limited number of high priority stories, which the product owner wants to be implemented first. Activities performed by student teams are united within the *development* column, which is further divided into the *analysis & design*, *coding*, *testing*, *integration* and *documentation* sub-columns. The *acceptance ready* column serves as a buffer between the development team and the product owner. Whenever a student team completes a user story, they move the corresponding card to the *acceptance ready* column, thus giving the product owner a sign to start acceptance testing. Then, the product owner pulls the card into the *acceptance* column and, if the user story passes all acceptance tests, moves it to the final column *done*.

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.

Figure 5 shows an example of a cumulative flow diagram. For each day, the number of cards in each column is displayed. The width of the area belonging to each board column shows how long a card stayed in that column, while the slope of the area belonging to the *done* column indicates velocity (i.e. number of items deployed per day).

In Figure 6, the calculation of lead time is shown. Bars represent user stories and the values at the top show the lead time in days. In addition to the total lead time, it is also possible to obtain the time that a card spent in each column.

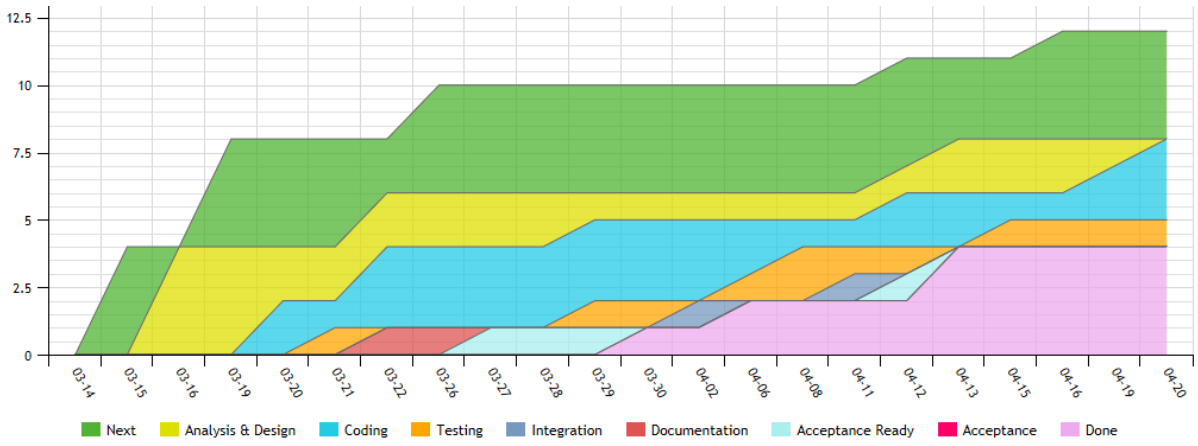

Figure 5: Cumulative flow diagram.

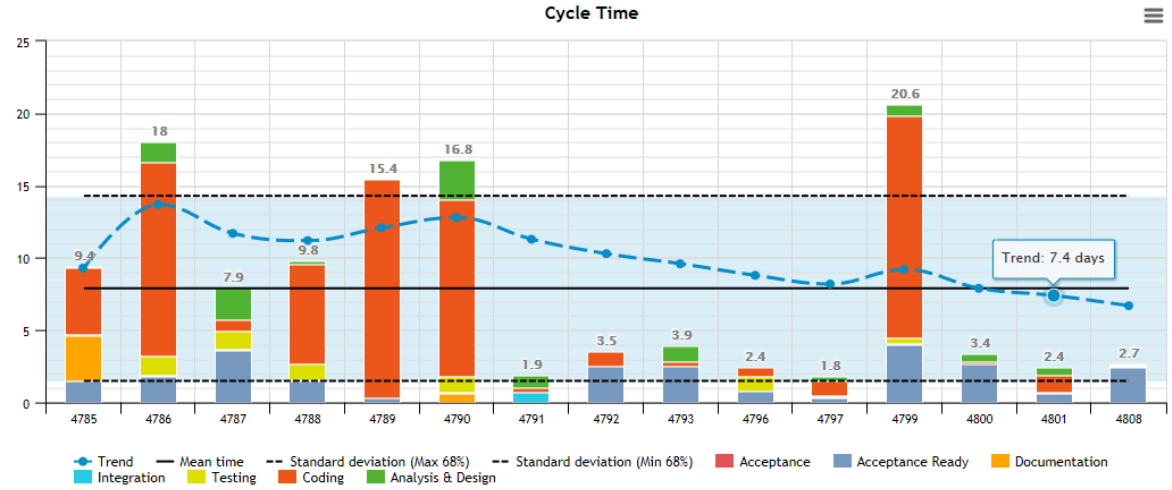

Figure 6: Lead time calculation.

CONCLUSIONS

Two software tools that support the management of students' SE capstone projects were presented. The first tool was developed at the University of Ljubljana and supports Scrum-based projects. The second one is a commercial tool and is used for managing Kanban-based projects. Both tools have proved to be useful and well accepted among students.

REFERENCES

1. Borman, D., Should software engineering projects be the backbone or the tail of computing curricula? *Proc. 23rd Conf. Software Engng. Educ. & Training*, Pittsburgh, PA, USA, 153-156 (2010).
2. Joint Task Force on Computing Curricula, Software Engineering 2014: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. IEEE CS and ACM (2015), 17 May 2018, https://www.computer.org/cms/peb/docs/se2014.pdf
3. Mahnič, V., The capstone course as a means for teaching agile software development through project-based learning. *World Trans. on Engng. and Technol. Educ.*, 13, **3**, 225-230 (2015).
4. Mahnič, V., A capstone course on agile software development using Scrum. *Proc. IEEE Trans. on Educ.*, 55, **1**, 99-106 (2012).
5. Mahnič, V., Teaching Scrum through team-project work: students' perceptions and teacher's observations. *Inter. J. of Engng. Educ.*, 26, **1**, 96-110 (2010).
6. Mahnič, V., A case study on agile estimating and planning using Scrum. *Electronics and Electrical Engng.*, 5, **111**, 123-128 (2011).
7. Mahnič, V. and Hovelja, T., On using planning poker for estimating user stories. *J. of Systems and Software*, 85, **9**, 2086-2095 (2012).
8. Azizyan, G., Magarian, M.K. and Kajko-Mattson, M., Survey of agile tool usage and needs. *Proc. Agile 2011 Conf.*, Salt Lake City, UT, 289-297 (2011).
9. 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).