Agile Software Project Management with Scrum

Viljan Mahnic, Slavko Drnovscek

University of Ljubljana, Faculty of Computer and Information Science
Trzaska 25, SI-1000 Ljubljana, Slovenia
viljan.mahnic@fri.uni-lj.si, slavko.drnovscek@fri.uni-lj.si

Abstract

The aim of our paper is to describe the experience we have using Scrum for agile software project management in a university environment. The paper is divided into three parts. In the first part an overview of the Scrum method is given. In the second part we describe how we used Scrum during the development of (a part of) the student records information system at the University of Ljubljana. Finally, the advantages of Scrum are summarized.

Keywords: agile methods, Scrum, software development

1 Introduction

Universities in Slovenia, like most universities in Central and Eastern European countries, usually develop their own software for university information systems. This is especially true for student records information systems which are often so specific that there are no commercially available solutions in the marketplace. The University of Ljubljana is no exception. In autumn 2001 a project was launched with the aim of building a comprehensive web based student records information system which enables remote access to data to all parties involved (viz. students, teachers, and administrative staff) [1, 2]. In the beginning, it was a pilot project of three departments, but after successful completion of the first phase, the project has been extended in order to serve other departments as well. At the moment of this writing the new student records information system is in use in 15 departments.[1]

Developing software for such a project requires a systematic and manageable software process. During the aforementioned project, we were using an adapted version of SSADM [3] for analysis and design, and Gantt charts for project planning. However, it often happened that the activities were late and some rework had to be done, partly due to changes in requirements and partly due to the fact that developers were not familiar with the development of web based information systems and the new tools they had to use (viz. Oracle Portal).

In order to make software development visible and adaptable we were looking for an agile approach to project management and decided to try Scrum [4, 5]. We first used Scrum to manage the development of the so called maintenance module that upgrades the aforementioned student records information system with facilities for the maintenance of data, administration of users, export of data to the data warehouse, import of data on freshmen students, etc.

The aim of this paper is to describe our experience using Scrum on this project. In the next section an overview of Scrum is given. In Section 3 we describe how we used Scrum during the development of the maintenance module and present some examples of Scrum artifacts, viz. a sample Product Backlog and Sprint Backlog. In Section 4 we summarize our experience with the use of Scrum.

2 Overview of Scrum

Scrum belongs to the family of agile software development methods that have attracted significant attention among software practitioners during last five years. Whereas the Extreme Programming method [6] that has been widely accepted as one of the most important agile approaches has a definite programming flavour (pair programming, coding standards, test driven development, refactoring, continuous integration), Scrum concentrates on managing software projects.

Scrum starts with the premise that software development is too complex and unpredictable to be planned exactly in advance. Instead, empirical process control must be applied to ensure visibility, inspection, and adaptation. The different environmental and technical variables (such as time frame, quality, requirements, resources, implementation technologies and tools, and even development methods) must be controlled constantly in order to be able to adapt to changes flexibly. This is achieved through an iterative and incremental development process.

[1] The university of Ljubljana has a peculiar organizational structure, permitting all departments a high level of autonomy. Consequently, each department maintains its own student records information system.

Scrum's skeleton is shown in Figure 1.[2] The lower circle represents an iteration of development activities that occur one after another. The output of each iteration is an increment of the product. The upper circle represents the daily inspection that occurs during the iteration, in which the individual team members meet to inspect each others' activities and make appropriate adaptations. Driving the iteration is a list of requirements. This cycle repeats until the project is completed.

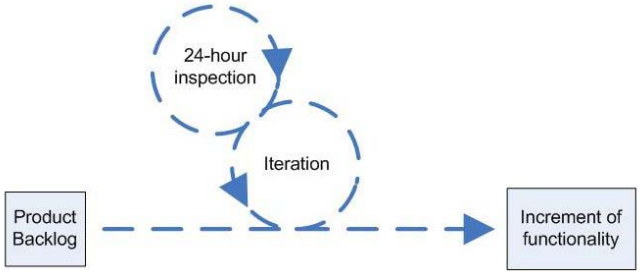

Figure 1 : Scrum skeleton

Scrum implements this iterative, incremental skeleton through three roles: the Product Owner, the Team, and the ScrumMaster.

- The Product Owner is responsible for representing the interests of everyone with a stake in the project and its resulting system. He maintains the Product Backlog, i.e., a prioritized list of project requirements with estimated times to turn them into completed product functionality.
- The Team is responsible for developing functionality. Teams are self-managing, self-organizing, and cross-functional, and they are responsible for figuring out how to turn Product Backlog into an increment of functionality within an iteration and managing their own work to do so. Team members are collectively responsible for the success of each iteration and of the project as a whole.

The ScrumMaster fills the position normally occupied by the project manager, but his role is slightly different. While the traditional project manager is responsible for defining and managing the work, the ScrumMaster is responsible for managing the Scrum process, i.e., for teaching Scrum to everyone involved in the project, for implementing Scrum so that it fits within an organization's culture and still delivers the expected benefits, and for ensuring that everyone follows Scrum rules and practices.

Detailed Scrum flow is shown in Figure 2. A Scrum project starts with a vision of the system to be developed. Then a Product Backlog list is created containing all the requirements that are currently known. The Product Backlog is prioritized and divided into proposed releases.

All work is done in Sprints. Each Sprint is an iteration of 30 consecutive calendar days. Each Sprint is initiated with a Sprint planning meeting, where the Product Owner and Team get together to collaborate about what will be done for the next Sprint. Selecting from the highest priority Product Backlog, the Product Owner tells the Team what is desired, and the Team tells the Product Owner how much of what is desired it believes it can turn into functionality over the next Sprint.

After deciding what has to be done in the next Sprint, the Team develops the Sprint Backlog, i.e., a list of tasks that must be performed to deliver a completed increment of potentially shippable product functionality by the end of the Sprint. The tasks in the list emerge as the Sprint evolves and should be divided so that each takes roughly 4 to 16 hours to finish.

Every day the Team gets together for a 15-minute meeting called a Daily Scrum. At the Daily Scrum, each Team member answers three questions: What have you done on this project since the last Daily Scrum Meeting? What will you do before the next meeting? Do you have any obstacles? The purpose of the meeting is to synchronize the work of all team members and to schedule any meetings that the Team needs to forward its progress.

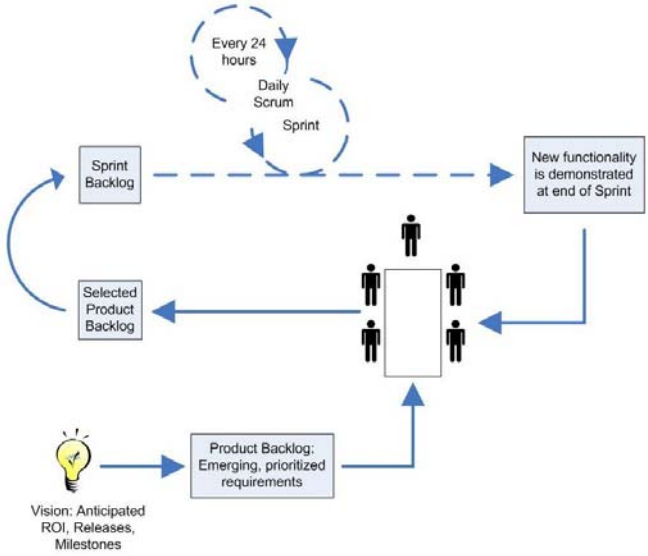

Figure 2 : Detailed Scrum flow

At the end of the Sprint, a Sprint review meeting is held at which the Team presents what was developed during the Sprint to the Product Owner and any other stakeholders who want to attend. After the Sprint review and prior to the next Sprint planning meeting, the ScrumMaster also holds a Sprint retrospective meeting in order to encourage the Team to revise, within the Scrum process framework, its development process to make it more effective and enjoyable for the next Sprint.

[2] Figures 1 and 2 are taken form [5].

Together, the Sprint planning meeting, the Daily Scrum, the Sprint review, and the Sprint retrospective constitute the empirical inspection and adaptation practices of Scrum.

3 Scrum at the University of Ljubljana

At the University of Ljubljana we started using Scrum within the project of building the so called maintenance module that enables the maintenance of all data required for the proper functioning of the student records information system, e.g., the maintenance of various code tables, installation parameters, lists of compulsory and optional courses for each study program, data about teachers of each course, etc. Additionally, the maintenance module provides facilities for administration of users and their rights, export of data to the data warehouse, import of data about newcomers, etc.

There were 4 people involved in the development of the maintenance module: three of them as members of the Team, and one who performed the roles of Product Owner and ScrumMaster. The roles of Product Owner and ScrumMaster are usually filled by two different persons, but in our specific situation the project was managed by the Head of Software Engineering Laboratory (viz. ScrumMaster) who was at the same time the Vice-Dean for Educational Affairs, thus representing also the interests of project stakeholders (viz. Product Owner).

After defining the initial Product Backlog and prioritizing requirements, it was estimated that the development of the required software will take three months. Therefore, three Sprints were planned as shown in Table 1.

The first four columns represent the Product Backlog item name, the initial estimate, the complexity factor, and the adjusted estimate. The complexity factor increases the estimate due to project characteristics that reduce the productivity of the Team. Initially, all Sprints were planned to take 20 working days; however, experience had shown that the adjusted estimate should be taken into account from the very beginning.

Table 1 : Product backlog at the beginning of the second Sprint

Backlog Description	Initial Estimate	Adjustment Factor	Adjusted Estimate	Work remaining until completion		
				Sprint 1	Sprint 2	Sprint 3
Defining a new user role	4	1.2	4.8	4.8	0	0
Maintenance of simple code tables	8	1.5	12	12	0	0
Maintenance of complex code tables	8	1.5	12	12	0	0
Sprint 1	20		28.8	28.8	0	0
Maintenance of complex code tables	5	1.1	5.5	5.5	5.5	
Maintenance of curriculum data	7	1.2	8.4	8.4	8.4	0
Maintenance of teacher's database	8	1.2	9.6	9.6	9.6	0
Sprint 2	20		23.5	23.5	23.5	0
Maintenance of installation parameters	3	1.2	3.6	3.6	3.6	3.6
Adding and removing users	3	1.1	3.3	3.3	3.3	3.3
Data import and export	5	1.2	6	6	6	6
Maintenance of data about degree pending students	2	1.2	2.4	2.4	2.4	2.4
Integration test	5	1.3	6.5	6.5	6.5	6.5
Installation	2	1.1	2.2	2.2	2.2	2.2
Sprint 3	20		24	24	24	24
Release 1	60		76.3	76.3	47.5	24

The Product Backlog evolved as the project evolved. Table 1 represents the situation at the beginning of the second Sprint. The rows correspond to the Product Backlog items, separated by Sprint and Release subheadings, and (within each Sprint) ordered according to their priority. All of the rows above Sprint 1 represent tasks that were worked on in that Sprint. The row Maintenance of Complex Code Tables is duplicated in Sprint 2 because it was only partly completed in Sprint 1, so it was moved down to the Sprint 2 for completion. The last three columns represent the Sprint during which the Product Backlog was developed.

After the Product Owner and the Team had agreed about the amount of work to be completed during each Sprint the Team was left alone to accomplish its tasks. Team members maintained the Sprint Backlog and participated regularly in the Daily Scrum meetings. Table 2 shows the Sprint Backlog of our project after the ninth day of the second Sprint. The rows represent Sprint Backlog tasks; the columns on the right represent the first 9 days of the Sprint. In each column, the estimated number of hours remaining to complete the task is shown for each task. If a task emerges later in the Sprint it is simply added together with the corresponding estimation (see task Export to different schemata). Obviously, in that case the total number of hours of work remaining may increase compared to the value of the previous day.

Table 2: Sprint Backlog after the ninth day of the second Sprint

Task Description	Responsible	Status (Not Started/In Progress/ Completed)	Hours of work remaining								
			1	2	3	4	5	6	7	8	9
Create SIFRANT_DATA table	Marko	Completed	2	0	0	0	0	0	0	0	0
Relocate icons for deletion and update	Marko	Completed	3	0	0	0	0	0	0	0	0
Relocate search and insert links	Marko	Completed	1	0	0	0	0	0	0	0	0
Develop toUpper function for Slovenian character set	Marko	Completed	3	3	0	0	0	0	0	0	0
Improve diagnostic messages	Marko	Completed	8	8	0	0	0	0	0	0	0
Use default sort order	Slavko	Completed	2	0	0	0	0	0	0	0	0
Include icons for scrolling up/down/left/right	Marko	Completed	1	1	1	1	1	1	1	0	0
Adapt sorting to Slovenian character set	Marko	Completed	3	3	3	0	0	0	0	0	0
Refactor sorting function	Slavko	Completed	8	8	8	8	8	8	0	0	0
Keep the selection unchanged if editing of a record is suspended	Slavko	Completed	2	2	2	2	2	2	2	2	0
Reset of sorting is not necessary	Slavko	Completed	2	2	2	2	2	2	2	2	0
Don't show the search criterion	Slavko	Completed	2	2	2	2	2	2	2	2	0
Refactor menu calls	Marko	Completed	4	4	4	0	0	0	0	0	0
Complete users manual	Marko	Completed	4	4	4	4	4	4	4	0	0
Export to different schemata	Marko	Completed							6	1	0
Maintenance of complex code tables			45	37	26	19	19	19	17	7	0
Print out curricula – selection criteria	Igor	Completed	10	6	5	0	0	0	0	0	0
Print out curricula – html	Igor	In Progress	16	16	16	16	8	2	2	2	2
Print out curricula – pdf, csv	Igor	In Progress	10	10	10	10	10	10	10	7	7

Copy STANJE table – previous year	Igor	Not Started	6	6	6	6	6	6	6	6	6
Edit STANJE table	Slavko	Completed	9	9	9	9	9	9	9	4	0
Create a list of courses for the next year based on data of previous year	Marko	Not Started	12	12	12	12	12	12	12	12	12
Add a course in the curriculum	Igor	Not Started	24	24	24	24	24	24	24	24	24
Delete a course from the curriculum	Igor	Not Started	6	6	6	6	6	6	6	6	6
Change course attributes	Igor	Not Started	8	8	8	8	8	8	8	8	8
Add a partial exam	Marko	In Progress	16	16	16	16	16	16	16	16	8
Delete a partial exam	Marko	Not Started	4	4	4	4	4	4	4	4	4
Change partial exam attributes	Marko	Not Started	6	6	6	6	6	6	6	6	6
Maintain exam prerequisites	Marko	Not Started	16	16	16	16	16	16	16	16	16
Maintenance of curriculum data			143	139	138	133	125	119	119	111	99
Print out all data for a specified year	Slavko	In Progress	12	12	12	12	12	12	12	12	6
Copy the last year settings	Slavko	Disregarded									
Add a new teacher	Slavko	Not Started	16	16	16	16	16	16	16	16	16
Delete a teacher for a specified year	Slavko	Not Started	16	16	16	16	16	16	16	16	16
Change examination dates because of teacher changes	Slavko	Not Started	12	12	12	12	12	12	12	12	12
Maintenance of teacher's database			56	56	56	56	56	56	56	56	50

There were no other elaborate plans during a Sprint, but the Team members were expected to be guided by their knowledge and experience on the basis of self-organization. Daily meetings allowed everyone on the project team to see the status of all aspects of the project in real time. The Sprint goals were an effective tool for keeping the Team on track and aware of expectations. We noticed an increase in volunteerism within the Team. They were taking an interest in each other's tasks and were more ready to help each other out. The best part of Scrum meetings was the problem resolution and clearing of obstacles. The meetings let the Team take advantage of the group's experience and ideas.

The ScrumMaster acted merely as a coach or mentor. His main task was to fend off changes during a Sprint to protect the Team from getting sidetracked. Nobody was allowed to change the Sprint goals. At the end of each sprint a Sprint review meeting took place at which the Team demonstrated the shippable increment of product functionality.

Each Sprint (except the first) produced tested and shippable code. During the first Sprint programs for maintaining code tables were developed that needed some elaborations (mainly regarding the user interface) requested by users at the Sprint review meeting. Users' remarks were considered in the second Sprint. After the second Sprint programs for maintenance of curriculum data and teacher's database were completed. Project was finished in time after the completion of the third Sprint.

4 Conclusions

The Team members found Scrum very useful. The use of Scrum improved the communication among them and maximized co-operation. It also increased their motivation and responsibility for the success of the project. On the other hand, it gave them freedom to maximally exploit their talent and knowledge during each Sprint. They were able to organize their work by themselves considering their preferences and special knowledge. At the end of the project they felt good about their job, their contributions, and they believed that they had done the very best they possibly could.

From the Product Owner's and ScrumMaster's point of view it was most important that the software development process became visible, controllable, and manageable. All impediments were immediately detected during Daily Scrum meetings and removed as soon as possible. It was also important that each Sprint produced a shippable increment of functionality that could be put into operation immediately.

Our experience corresponds to findings reported in the literature [7, 8]. As a result of the use of Scrum:

- the product becomes a series of manageable chunks,
- progress is made, even when requirements are not stable,
- everything is visible to everyone,
- team communication improves,
- the Team shares successes along the way and at the end,
- customers see on-time delivery of increments,
- customers obtain frequent feedback on how the product actually works,
- a relationship with the customer develops, trust builds, and knowledge grows, and
- a culture is created where everyone expects the project to succeed.

References

[1] Mahnic, V., Bajec, M. Introducing e-business technology in the area of student records, Proceedings of the 8th International Conference of European University Information Systems EUNIS 2002, Porto, Portugal, 19-22 June 2002, pp. 300-304.

[2] Mahnic, V., Bajec, M. Reengineering of the student records information system, Proceedings of International Conference on University Information Systems UNINFOS 2003, Nitra, Slovakia, 3-5 September 2003, pp. 212-218.

[3] Goodland, M., Slater, C. SSADM, A Practical Approach, McGraw-Hill, 1995.

[4] Schwaber, K. and Beedle, M. Agile Software Development with Scrum, Prentice Hall, 2002.

[5] Schwaber, K. Agile Project Management with Scrum, Microsoft Press, 2004.

[6] Beck, K. Extreme Programming Explained, Addison-Wesley, 2000.

[7] Rising, L., Janoff, N. S. The Scrum Software Development Process for Small Teams, IEEE Software, July/August 2000.

[8] Sutherland, J. Agile Development: Lessons Learned from the First Scrum, October 2004, http://jeffsutherland.com/scrum/FirstScrum2004.pdf, page visited on 21.2.2005.