Scrum and Agile Methods in Software Engineering Courses

Jennifer Campbell
University of Toronto
campbell@cs.toronto.edu

Stan Kurkovsky
(moderator)
Central Connecticut State University
kurkovsky@ccsu.edu

Chun Wai Liew
Lafayette College
liewc@lafayette.edu

Anya Tafliovich
University of Toronto
Scarborough
atafliovich@utsc.utoronto.ca

Categories and Subject Descriptors

• Software and its engineering → Agile software development.
• Social and professional topics → Software engineering education

Keywords

Scrum; Agile software development; Software engineering

1. SUMMARY

Agile software development has been formally introduced in 2001 in the "agile manifesto" [4]. Agile methods are best described as an amalgamation of four core principles describing lightweight iterative software processes:

- Individuals and interactions over processes and tools;
- Working software over comprehensive documentation;
- Customer collaboration over contract negotiation;
- Responding to change over following a plan.

Over the years, a number of agile frameworks, such as Extreme Programming and Scrum, have evolved and matured. The underlying philosophy of Scrum recognizes that the customers often change their mind about the product they want and that the development challenges are unpredictable by their nature. Consequently, Scrum embraces the fact that the problem being solved cannot be fully understood or described from the start. Instead, Scrum focuses on maximizing the ability of the development team to quickly deliver in response to emerging requirements.

The aim of this panel is to present the experiences and challenges of practicing Scrum and agile methods at a variety of computer science programs.

2. OUR EXPERIENCE

This section describes the experience of each panelist (in alphabetical order).

2.1 Jennifer Campbell

At the University of Toronto, we offer a second-year course [1] in which we teach the fundamentals of software design and give students an opportunity to apply what they are learning in designing and developing Java and Android applications. Students are also introduced to the basics of software engineering and follow a prescribed Scrum-like software development process as they complete a team project. Each semester, between 200 and 400 students enroll in this course.

The software development process involves an instructor generated project feature set, multiple software design and development phases, subsets of the feature set for each project phase, planning meetings, status meetings, and post-phase peer- and self-evaluations. These loosely correspond to key Scrum elements including product backlog, sprints, sprint backlog, sprint planning, daily scrums, and sprint retrospective. We introduce students to a variety of software tools to support their project, including version control (Subversion) and a fully featured IDE (Android Studio). We also use the CATME web tool (www.catme.org) for team formation and team member evaluations.

For most students, this project is their first team software development project and by introducing this Scrum-like process we aim to keep the multi-week multi-phase team projects on track. Exposing students to a software development process at this stage of their academic program also allows us to foreshadow the third-year software engineering course that involves an agile software development project.

2.2 Stan Kurkovsky

At Central Connecticut State University students have at least two opportunities to experience Scrum.

Students may choose to get an in-depth experience with Scrum in a game development course, which is based on the book written by one of our graduates [2]. Typically, students take this course immediately after Data Structures and have a limited experience with software development processes or working as a part of the team. This course introduces students to the core principles of Scrum using a broad range of agile games, some of which are based on those from tastycupcakes.org. Students learn about the Scrum roles; sprints and their planning, reviews, and retrospectives; product backlog, user stories and their prioritization.

A few weeks into the course, student teams are ready to begin applying Scrum in a game development project lasting until the end of the semester. The course is structured so that there is enough time for at least three 2-week sprints. The instructor plays

SIGCSE'16, March 2–5, 2016, Memphis, TN, USA.
ACM ISBN: 978-1-4503-3685-7/16/03.
DOI: http://dx.doi.org/10.1145/2839509.2844664

the role of the Scrum Master with all sprint planning and review meetings taking place during the class time in the second half of the semester.

As a part of a required formal senior Software Engineering course, all students in our program are introduced to Scrum in a 3-hour long LEGO-based simulation [5], which prepares them for an agile course project that focuses on short iterations with evolving requirements.

2.3 Chun Wai Liew

At Lafayette College students take a second year course in which we teach the fundamentals of software engineering and students learn about the basic tools and techniques involved in developing group projects [3]. This course serves two purposes in our curriculum. Firstly, it teaches the students about the software development tools that they will need in the upper level courses -- software repositories, time management/planning/scheduling tools, weekly status and bug reporting tools. Secondly, they learn the basics of the agile process - (1) repeated short iterations, (2) unit testing from the beginning, (3) interleaving of specification development, prototyping and design.

For our students, this is (a) their first group development project, and (b) their first "medium-sized" project. They learn how to keep the projects on track (most of the time) and also how to coordinate their activities as a group. This experience allows us to have richer projects in the upper level courses and also to offer a capstone software engineering course with external clients.

We also have a second software engineering course (the capstone) where students apply the scrum/agile methodology that they have learned on a project with external (to the college) clients.

2.4 Anya Tafliovich

At the University of Toronto Scarborough, we have a sequence of three courses that involve aspects of Agile software development, to various degrees. The first in the sequence is a second-year course described above. The students who complete this course have had experience with a 6-7 weeks long team project, in which they follow a fairly rigidly prescribed Scrum-like simplified development process. They are familiar with the ideas of source control (subversion), iterative development (prescribed project phases), backlogs (feature lists for each phase, prescribed by the instructor), and the different kinds of meetings involved.

The second course in the sequence is a third-year Introduction to Software Engineering course. In this course the students are introduced (in lecture) to all aspects of Agile development; we look at Scrum and XP in detail. The course has a 10-week long team project, in which the students have hands-on practice with all aspects of Agile development. We do not follow either Scrum or XP religiously, as we found this to be counter-productive in a course setting. Instead, we have developed an Agile hybrid process suitable for third-year computer science student teams, in a large class (we expect over 20 teams in the coming term). We bring in a real client, with a real need for software. The client is non-technical (a member of a different department), which gives the students practice with the challenges of requirements gathering. The students practice writing user stories, producing release plans and iteration plans, using state-of-the-art tools that support Agile development (GitHub, online project management tools, etc.), and following through with iterative development, re-planning and re-estimation as needed, until the final product is ready for the final presentation to the client at the end of the term. Crucial to the success of the course are weekly deliverables and weekly meetings with the team TA—these keep the projects on track.

The third course in the sequence is a fourth-year course, in which the student teams participate in a large open-source project hosted on GitHub. If there is enough interest, we can elaborate on this course and the use of Agile methods in it during the panel.

3. CHALLENGES

The panelists are also planning to have an open discussion focusing on lessons learned, challenges, and their possible solutions:

- Why should we teach Scrum/Agile?
- What are the biggest obstacles to adopting Scrum/Agile in teaching Software Engineering?
- How to fit Scrum process into a class schedule? (daily meetings are usually impossible)
- Who plays the roles of the Scrum Master and Product Owner?
- Are two courses necessary to introduce Scrum/Agile? Can the approach work if the students only have one experience? What carries over from the first course to the second?
- What aspects of the Scrum/Agile process are most difficult for the students to master?
- Is there a difference in teaching Scrum/Agile to first/second year students as opposed to fourth year students?

4. REFERENCES

[1] Campbell, J. and Tafliovich, A. 2015. An experience report: using mobile development to teach software design. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (SIGCSE '15). 506-511.

[2] Keith, C. 2010. *Agile Game Development with Scrum*. Addison Wesley.

[3] Liew, C. W. 2005. Teaching software development skills early in the curriculum through software engineering. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (ITiCSE '05). ACM, New York, NY, USA, 133-137.

[4] Manifesto for Agile Software Development. 2001. http://www.agilemanifesto.org/

[5] Paasivaara, M., Heikkilä, V., Lassenius, C. and Toivola, T. 2014. Teaching students scrum using LEGO blocks. In *Companion Proceedings of the 36th International Conference on Software Engineering* (ICSE '14). ACM, 382-391.