Running an Agile Class

Kevin A. Gary
Arizona State University
7171 E. Sonoran Arroyo Mall
Mesa, AZ 85212
kgary@asu.edu

Abstract

Agile methods, particularly Scrum, apply an empirical process model to the complexity of software development. The reasoning, in short, is that a highly iterative process with specific short-term goals can be instrumented to provide constant feedback to change and disruption. Agile methods focus on short iterations, adaptive work assignments, constant feedback, and process visibility to address the fundamental nature of complexity. The author is currently experimenting with agility in the classroom through the incorporation of these mechanisms and the support of online tools. Online tools can help in several ways, such as helping the instructor and teaching assistants scale up the agile teaching process. Initially we have found that the most impactful of online tools is a scrum taskboard, or *Scrumboard*, to make the work of the class visible to all stakeholders. In the context of the technology-supported classroom, the online Scrumboard not only helps the instructor and teaching assistants (yes, the *pigs*) manage the busy work of running the course, but lets the stakeholders (students, or *chickens*) understand what is happening now and next, and how their progress influences what happens on the board. This paper will focus on the use of an online Scrumboard in an upper-division project course, sharing the instructor's experience combined with a survey completed by the students. It will expand on agile teaching and its potential for managing today's complex higher education class.

Introduction

This is **not** another paper describing how to teach agile methods to software engineering students. Rather, it is a paper on experiences using an agile methodology to execute a college course. This paper shares instructor experiences and student evaluations of running a course as a series of scrum *sprints*, and sharing the visibility of the sprints via an online *scrumboard*.

The higher education landscape is changing in many ways; online courses, project-based learning, active/discovery-based learning, technology aids, high articulation rates, and so on. The "sage on the stage" teaching model is as obsolescent as the waterfall model in software engineering. Instructors are now expected to balance a fast-paced classroom incorporating new learning models and complex course plans with the events that inevitably arise when dealing with nonlinear processes. In other words, dealing with the complexity of teaching material while guiding self-directed activities to teams of students with different educational backgrounds.

The experience described in this paper is from an upper-division project-centric course where student teams work on a project while learning and applying new concepts. The author found that constantly revising a course plan based on student progress was a losing game; there were too many events that altered the plan. The "instructor as coach" model was difficult to scale,

students were confused between the learning process and the project process, and too much time was spent tweaking the course *with the students* than was spent on executing the learning process.

The author's epiphany was realizing an inability to control a predictive (course) plan and instead applying a reactive, empirical process model. Agile methods, notably Scrum, provide mechanisms for implementing empirical process control. *Sprints*, *sprint goals*, and *continuous feedback* provide for directed process execution with the ability to adapt to change [6]. In terms of a course, a sprint is a unit of time; a sprint goal is the outcome (or suboutcomes) to be achieved by the sprint, and the class meeting time, formative and summative assessment activities, and technology-based touchpoints (email, project dashboards, discussion forums) a form of continuous feedback. This paper describes the agile mechanisms put in place and shares the results of a small survey of class participants on the utility of the approach. Some thoughts on progressing the idea of agile classrooms are provided at the conclusion of the paper.

The Benefits of Agility in a Course Context

Agile methods are, of course, the silver bullet software engineering has long awaited. It provides productivity, innovation, creativity, time-to-market, under budget costs and zero learning curve to all adopters on their software projects. Plus, it cures the common cold. So why shouldn't we shoot that bullet at the difficulties in the changing landscape of higher education? After all, the modern classroom is distancing itself day by day from the classrooms of yore. Learners arrived with predetermined expectations of what energy they should and will expend on the class. Technology is as much an intrusion as an innovation; we swing the Mjölnir wildly with tablets, cell phones, content delivery systems, online interactive collaboration and of course MOOCs (my how the mighty have fallen…). New pedagogies exuberantly attack the sage on the stage with enough variants to start their own revolution (oh, wait…).

Facetiousness aside, Agile is in the category of hyped technologies and some are starting to track its spiral into the "Trough of Disillusionment" [4,7]. Most critics do not fault agile methods themselves, but rather the inexperienced or inaccurate application of agile methods by uninformed middle management. Of course over time agile methods will mature into better-understood variants and adaptations and appears to be here for the long haul [5]. It is the author's experience that Agile experienced an initial popularity as part of a developer revolt of sorts; emboldened by the success of open source software and community development spurred by the Internet, developers rapidly adopted (sometimes in secret) agile means of executing their tasks. Traditional project and product managers did not always buy into these methods, as they did not understand them, and felt more comfortable with traditional project management techniques and tools, and needed more long-term guarantees than agilist developers would provide (ignoring the fact that these guarantees only guaranteed failure).

Much of the agile literature espouses the wonders of agile from the software engineering perspective, but there are also several business-oriented benefits that have helped agile overcome initial project/product management skepticism [3]. Most notable of these are an ability to adapt to changing requirements, expectations, and operating conditions, an ability to cross functional boundaries, and the transparency and visibility of the process to all stakeholders. While there are

many other benefits of agile, it is these benefits that convinced the author that perhaps *agile* applied to *teaching and learning* could address some of the issues arising from the evolving education landscape. Specifically:

- The changing environment in which the modern university course is taught is changing as mentioned above. Changes in student expectations, changes technology, changes in pedagogy map to the changes in requirements and operating conditions that agile addresses for software development organizations. Can agile help manage this change for the educator?
- More and more (at least at the author's institution), higher education students are encouraged to work in multidisciplinary teams and even define custom multidisciplinary degree programs. At the author's institution students may cluster 4 courses into a *secondary focus area* combining a student's interests with degree program outcome requirements. The result from the faculty perspective is more students showing up in upper division classrooms with a variety of backgrounds and interests. This maps to the agile benefit of dealing with cross-functional boundaries.
- Transparency and visibility of the process is a tremendous impetus for agile adoption on software development projects. Project managers enjoy being able to receive constant feedback on projects via daily standup meetings and dashboards, while product managers and clients gain visibility and control into the management process via short iterations and frequent planning sessions. Risks are identified early and addressed through a scale-down / scale-up set of mechanisms. To the author this is the main potential benefit of applying an agile approach to teaching and learning. Providing students with visibility into where the course is, what learning goals are being achieved (in the current iteration), and getting constant feedback provides them a context and organization of the learning process, resulting a less frustrated and more engaged student participant.

The next section will present the specific agile mechanisms put in place in an upper division project-centric course in the spring of 2013, with the focus on achieving the third agile benefit of transparency and visibility.

Agile Practices for Course Management

The traditional means by which a higher education instructor manages her/his course offering is to distribute a planning document, or *syllabus*, on the first day of class. This syllabus is usually an example of a prescriptive process model – it contains process steps (course topics and perhaps a calendar), process constraints and resources (textbooks, homework submission policies), process assessment (learning outcomes and grades), and policies for resolving process errors (grade appeals, academic integrity policies, etc.). While usually prescriptive there is nothing mandating that it be constructed as such; one can argue that the most important content of the syllabus are the process assessment components, as they inform the student what s/he will learn and how feedback will be provided to ensure the process of learning in the classroom supports achievement of those outcomes. Nonetheless most courses map out a syllabus and use it as a plan to prescriptively execute throughout the semester. Technology only reinforces this approach, as many course management systems now provide a syllabus tool and allow the instructor to generate learning activities from that tool, and often map them onto a course calendar.

But what if the instructor focused less on course content and learning activities and more on learning outcomes, continuous feedback, and making in-time adjustments to learning process execution during the semester? Then s/he starts to sound agile. To realize this approach, the author introduced some agile mechanisms based on the Scrum methodology [6] in a junior level project-centric course in the spring semester of 2013. The course material focused on software development best practices, though that is not particularly important outside of there being specific measurable technical competencies as part of the course outcomes. 23 students were enrolled in the course, 22 of the 23 passed the course.

The principal agile concept introduced was the Scrum *sprint* with accompanying *product* and *sprint backlogs*. Using the Scrumwise tool, obtained via a free academic license, the instructor created a product backlog by starting from the course learning outcomes. From there, eight sprints were mapped out over the course of the semester corresponding to eight modules the instructor wished to cover. Figure 1 shows a completed Scrumboard from Scrumwise.

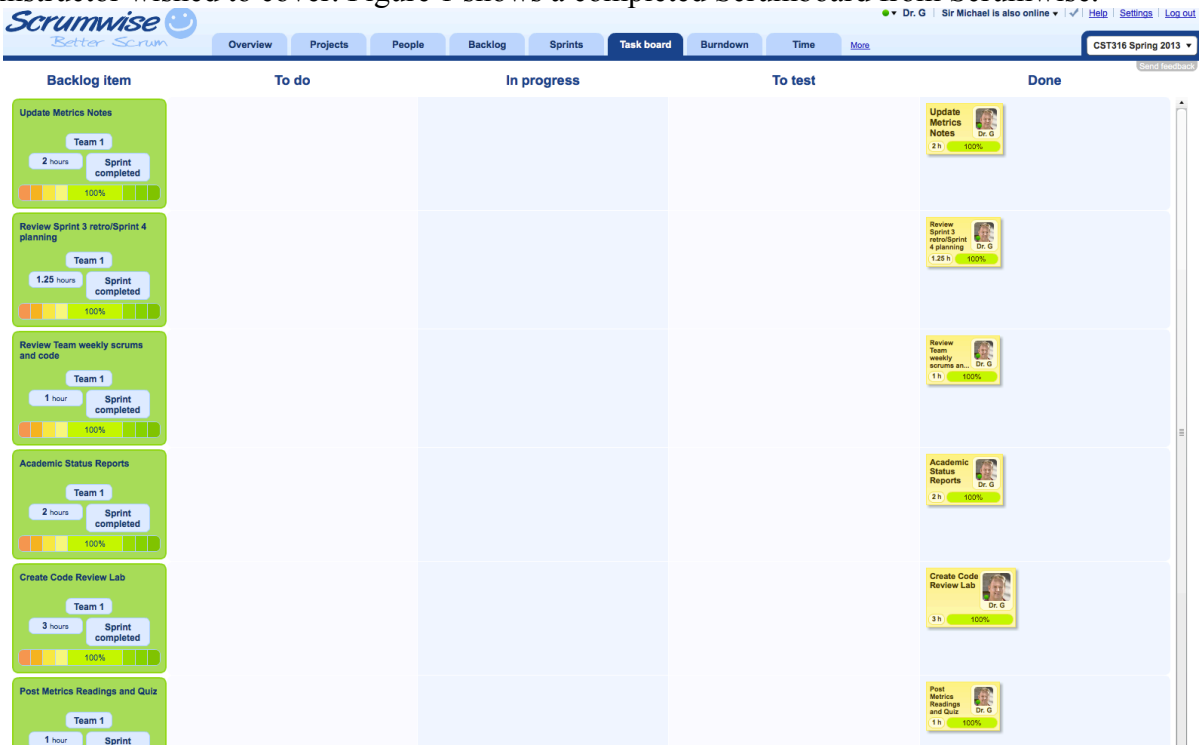

Figure 1. Completed Scrumboard from a course sprint

The notion of a sprint is fundamental to the introduction of agile teaching and learning. First, the sprint provided a time-boxed way to achieve a learning outcome within a course, or at least to cover some material associated with a learning outcome. Second, the concept of a *sprint goal* is very important to this process. A sprint goal in Scrum is the specification of what the team is trying to achieve by completing the sprint. It is best used as a litmus test to determine whether the sprint was successful or not – instead of assuming a sprint is successful merely by completing all the tasks associated with a backlog item. This is an important, oft-overlooked aspect of Scrum, and one that enables it to react to change so well. The focus of a sprint is on achieving a goal, not on completing activities. Likewise, for a teaching and learning process, the focus should be on understanding whether students are achieving learning outcomes and not as much

on whether they actually (or the instructor actually) completes various tasks. Finally, the Scrumboard provides a powerful *information radiator* [2] for the class. It provides constant feedback as to what the instructor is doing, what is coming next in the class, what is changing, and what the goal is.

The Scrumboard for the course was made available throughout the course by adding the students as stakeholders ("chickens" in Scrum lingo) while the instructor and teaching assistants were the owners ("pigs"). Anecdotally, the author found that utilizing the Scrumboard and organizing course tasks in sprints was a great way to manage the constant change and minutiae of executing a course. Interestingly, sharing the Scrumboard with students appeared to be a step toward better communication of weekly course goals and activities, resulting in less communication time around where the class was and what was coming next [1]. Of course this is the perspective of a single instructor. A survey was also conducted asking the students for their perspectives on utilizing Scrum in the class. The survey asked 3 simple rating questions and one qualitative free-form type of question. Of the 23 enrollees, 13 responded to the survey.

Q1: "I liked being able to see what the instructor was preparing next for our XXX class"

#	Answer		Response	%
1	Strongly Agree		6	46%
2	Agree		5	38%
3	Neither Agree nor Disagree		2	15%
4	Disagree		0	0%
5	Strongly Disagree		0	0%

Table 1. Results from survey question Q1

Q2: I checked the XXX Scrumboard periodically throughout the semester

#	Answer		Response	%
1	Strongly Agree		2	15%
2	Agree		7	54%
3	Neither Agree nor Disagree		1	8%
4	Disagree		1	8%
5	Strongly Disagree		2	15%

Table 2. Results from survey question Q2

Q3: I would recommend the Scrumboard for the next offering of XXX

#	Answer		Response	%
1	Strongly Agree		6	46%
2	Agree		5	38%
3	Neither Agree nor Disagree		2	15%
4	Disagree		0	0%
5	Strongly Disagree		0	0%
	Total		13	100%

Table 2. Results from survey question Q3

(Note: XXX is used to hide the course identification)

Q1 directly asks whether students found the Scrumboard useful, and most students responded positively. Likewise Q3 is similarly positive, though a cross-tabulation of the responses reveals

that respondents did not answer these 2 questions the same way, despite the identical aggregate results. Q2 responses were more distributed, though 9 of 13 answered positively while only 3 of 13 answered negatively. The cross-tabulation reveals that for those 3 that answered negatively to Q2 (i.e. they did not check the Scrumboard periodically), 2 of those 3 answered Q1 and Q3 positively. The author believes this is because a postmortem reviews of the class Scrumboard at the end of the semester led to observations that some students did not track the board closely as they did not fully understand the intent of these sprints in the first place.

Q4: Please enter any thoughts or recommendations you would make regarding your experience using a Scrumboard by the Instructor in XXX

There were only 7 responses to this question, and most were comments on the Scrum tool employed and not on the Scrumboard itself. That is, as often happens with open-ended class style evaluation surveys, the students riffed on what they wanted, not what was asked! One exception:

> *"It provided a good reference for where we were in the class and what to expect. But I did not use it as much as I probably should have."*

Which the author believes represents the typical perspective of students in the class.

These results are taken from a single semester offering with a relatively small number of participants in a single course section, so the author is aware of the limitations of drawing conclusions from the survey. Further the survey asks about perspectives on learning and the utility of a teaching and learning tool, and does not directly measure learning outcomes. Future iterations require a measurement vehicle where the impact of the methodology and tools on learning outcomes is directly addressed.

Discussion and Future Directions

This is a new way of running a class, and communicating the model and the mode of work to the stakeholders (the students) is critical to its success. Confounding this challenge was the fact that students were also executing their own sprints for their class projects. While at first the author thought this would help minimize the learning curve for conducting an agile class, it appears to have confused the matter for some students (as evidenced by Q2).

The careful reader has probably noticed by now that the definition and execution of the sprints does not exactly match the motivational philosophy given for employing sprints above. Most of this is due to the simple nature of trying this in a real class the first time, and the practical obstacles that arise. For example, eight sprints were defined because there were eight pre-existing curricular modules available for the course based on previous course offerings. It was thought that a straight one-to-one mapping of modules to sprints would be the least intrusive way to introduce agility. But, the results were sprint goals narrowly defined by technical competencies (e.g. "the student will be able to construct a unit test in tool X") and not easily mapped to course level outcomes (e.g. "at the conclusion of this course, students will understand the principles of quality source code development, the challenges in introducing best practices on software development projects, and apply tools and techniques for achieving code quality in the

context of a scalable project"). The author had to create a suboutcome layer and map these suboutcomes, representing specific technical competencies to the course outcomes, and all this was tracked outside the sprints. For the next iteration of the course in Spring 2014, the author is considering how to introduce *epics* as a means to manage *themes* that corresponding to learning outcomes. For example, one epic might represent quality coding practices with sprints dedicated to static analysis, refactoring, and code reviews (3 example modules from the course) while another epic might represent software testing and include sprints on unit testing and continuous integration and testing (2 more modules). Other approaches may work as well.

Despite the infancy of this research, some clear benefits have been identified. Transparency, visibility, and organization are powerful vehicles for progressing learner motivation and outcomes. This is inline with larger studies on these basic mechanisms (a good summary recently published here [1]). The nature of change may be distinct from software engineering, but the transformation of the modern higher education experience means that change does occur and takes many forms, and perhaps *empirical process control* is one mechanism to help the overwhelmed instructor adapt. Of course not all is good; there is danger is adopting a process model from one domain to another domain of practice, though we note ironically this is what happened with Scrum in the first place. Many of the drawbacks encountered so far are likely due to the infancy of the work combined with this danger. Figuring out the right agile practices, defining a methodology for introducing those practices in the classroom, and assessing their impact requires deeper investigation and rigor, and this is the trajectory of future work.

Finally, the author notes that this work is part of a larger research impetus to investigate and implement strategies taken from engineering and computing disciplines in the context of teaching and learning processes. Software engineering in particular has seen great change in the understanding of best practices, driven by new process models like agile but also from the expansion of community-oriented development spurred by the Internet. In short, software engineers are becoming expert practitioners in working on problems with high rates of change – in requirements, methods, tools, technology, and expectations. That sounds a lot like what is starting to happen in higher education, so perhaps we should look to process and practices we have right under our nose for the solution!

References

1. Barrett, D., "Teaching Clearly Can Be a Deceptively Simple Way to Improve Learning", *The Chronicle of Higher Education*, November 22, 2013.
2. Cockburn, A. (Online), "Information Radiator", Available at http://alistair.cockburn.us/Information+radiator, (last accessed January 4, 2014) June 19, 2008.
3. Highsmith, J. and Coburn, A., "Agile Software Development: The Business of Innovation", *IEEE Computer* September 2001, pp. 120-122.
4. Janes, A., Succi, G., "The Dark Side of Agile Software Development", Proceedings of the Third ACM conference on Systems, Programming, Languages and Applications: Software for Humanity (SPLASH 2012), Tucson, AZ. October 2012.
5. Kajko-Mattsson, M., Aguiar, A., Boness, K., Kaindl, H., Pooley, R., and Tael, A., "Long-Term Perspective of Agile Methods", Panel report in the Proceedings of the 4th International Conference on Software Engineering Advances, 2009 (ICSEA '09), Porto Portugal, September 2009.
6. Schwaber, K. and Beedle, M., *Agile Software Development with Scrum*, Prentice-Hall 2001.
7. Wilson, N. (Online), "The Trough of Disillusionment" Available at http://blogs.gartner.com/nathan-wilson/the-trough-of-disillusionment/, (last accessed January 4, 2014) July 12, 2012.