https://doi.org/10.48009/1_iis_2016_119-127

Issues in Information Systems
Volume 17, Issue I, pp. 119-127, 2016

ADDING AGILITY TO THE MIS CAPSTONE

Thom Luce, Ohio University, luce@ohio.edu

ABSTRACT

The use of agile approaches to software development has grown rapidly over the last two decades while colleges and universities have struggled to keep up with the change and prepare students for the working environment they will be entering. This paper describes one attempt to introduce agile methodology, specifically Scrum, into a capstone MIS class. The course first introduced Scrum through a series of readings, videos and short training sprints and then used Scrum to manage a client-based project with between 6 and 8 teams working on different aspects of the project. The paper describes some of the outcomes of four semesters worth of projects, including client and student reactions to the overall project and the use of Scrum along with areas for future research.

Keywords: Agile, Scrum, MIS Capstone, Systems Development

INTRODUCTION

Software developers have been working on alternatives to the traditional waterfall software development methodology for years but a major high point in this effort occurred in 2001 with the publication of the Manifesto for Agile Software Development (agilemanifesto.org, 2001). Agile software development simplifies the software development process by dividing projects into small chunks that can be coded, tested and delivered in a short period of time, typically a few weeks. Agile approaches accept the impossibility of knowing all requirements at the beginning of the project and welcome changing requirements as they become better known and understood. Since the publication of the Agile Manifesto acceptance and use of agile techniques have steadily grown in organizations. In a recent survey by VersionOne (VersionOne, 2016) 95% of the respondents said they were using at least some agile development techniques with 33% indicating that they had been using agile for more than five years and 30% using it for between three and five years. The same survey revealed that 58% of the respondents were using Scrum (Schwaber and Beedle, 2001, Sutherland, 2014, Rubin, 2012) while 10% were using a hybrid of Scrum and extreme programming (XP) (Beck & Andres, 2014).

Scrum works with teams consisting of three main players; the product owner who creates and maintains the list of work (called the product backlog), the scrum master who helps keep the team focused on its goals and protects them from outsiders and the development team itself. Scrum divides the work into a series of time boxed units (typically 2-4 weeks) called Sprints and allows for very few changes during the execution of the sprint (Scrum Alliance, 2016). XP typically works in shorter time units (one or two weeks) and welcomes changes at any time. There are also differences in the order of work done during the execution cycle with Scrum giving the development team more flexibility than is normally available to an XP team (Cohn, 2016).

According to the 2015 VersionOne report 87% of responding companies view the ability to manage changing prior priorities as the number one reason for using agile. In the same report 85% of respondents rated increased team productivity as a top benefit while 84% sited improved project visibility.

While industry has been relatively quick to embrace agile and Scrum, higher education has lagged somewhat behind. Tan, Tan, & Teo (2008), Devedžić, V. & Milenković, S.R. (2010), Lu & DeClue. (2011), Mahnic (2010) and others discuss implementation of agile in both undergraduate and graduate courses in computer science and software engineering and Bair & Riggins (2012) discuss a hybrid methodology (Waterfall with Scrum) used in a capstone CIS class. While there have been reported attempts to implement agile and Scrum in CS and SE courses, and to a lesser extent CIS and MIS courses, Soundararajan, Chigani, and Arthur (2012) point out the minimal emphasis on agile software development in many colleges and universities.

THE MIS PROGRAM

The MIS major at Ohio University changed from an eight-course sequence to a six-course sequence when the university underwent a conversion from quarters to semesters (Luce, 2010). The current major begins with an introduction to systems analysis and design that discusses the traditional Waterfall model along with various approaches to agile, including Scrum. The second introductory course covers database management and business intelligence and introduces the student to SQL in Microsoft's SQL Server, a little program access to data in Visual Studio Express and several BI and visualization tools such as Tableau.

The second set of courses includes a software development course that introduces C# in the Visual Studio Express environment. In addition to basic programming skills the course introduces and expands on data access using existing controls and code. The second course in this tier introduces the student to enterprise systems using tools available through the SAP Academic Alliance.

The final tier of courses incudes a capstone concepts course and the capstone development course, the course that is moving to Scrum. This class was traditionally a live client based course taught using a modified waterfall, or spiral development approach. Based on feedback from the employers of our students and our MIS advisory board, we started moving the course to agile and Scrum in the fall of 2014. This paper is not a formal research paper but describes early efforts to convert the MIS capstone course to Scrum. The paper looks at how we attempted to apply Scrum roles, artifacts and Scrum methodology in a senior level MIS capstone class that meets three times a week for only 55 minutes per class.

COURSE STRUCTURE – FIRST YEAR

Phase One

The capstone systems development course is structurally divided into two phases. During the first phase students work through a series of exercises designed to provide a programming refresher and to introduce new tools and procedures that could prove useful in the second phase of the course. The second phase is a live-client project typically involving creation of an interactive website.

In the Scrum version of the course the first phase is divided into a series of short *training sprints* lasting one week each. For these sprints the instructor served as both the *Scrum Master* and the *product owner* (see Rubin, 2012 for a description of terms). *User stories* were created for the exercises previously described by Luce (2014), added to a *product backlog* and prioritized.

At the beginning of each training sprint one or two user stories were moved to the *sprint backlog* with the instructor providing the definition of done for each user story requirement. For example, the first sprint requires the development teams to select a standards based cascading style sheet (CSS) website template, convert it to an ASPX master page and convert menu items from HTML anchor tags to ASPX Hyperlink controls. The requirement is "done" if the master page functions in the .NET environment and the menu items work both locally and when the web site is moved to a public server. During the *sprint review* any team's work that did not meet the definition of done was returned to the team's project backlog. Grades were not assigned to work until it met the definition of done.

At the end of each training sprint the teams were required to meet for a brief *retrospective* where they discussed three questions: What went well during the sprint? What went wrong or didn't work well? What can they do to improve things for the next time around? (Scrum Alliance, 2014)

Because students had very little exposure to Scrum before beginning the first phase of the course they had to learn about it in a just-in-time process. This was accomplished through a series of readings from Rubin's book (Rubin, 2012), from the Scrum Alliance (2015) and Scrum.org (2015) and others. They were also assigned a series of

videos covering topics related to the assigned readings from YouTube (2015) the Scrum Training Series (2015), SCRUM01-SCRUM10 (2015) and others.

Phase Two

The second phase of the class began with the client (product owner) coming to the classroom, talking about their problem, presenting relevant documents and holding a question and answer session. Sessions were recorded and posted so students could review them as needed. After the client finished the students were asked to think about potential user stories embedded in the presentation and to be prepared to talk about them at the next class.

During the follow-up class students were handed a stack of 3x5 Post-it® Notes and asked to write down one user story related to the project per page. They were instructed to use the three-part user story format (Agile Modeling, 2015) discussed in class: As a <role> I want <something> So that <benefit>.

After writing the user stories the class discussed the number of roles that had been defined and attempted to clarify any ambiguity in role names. Once the set of roles was agreed upon the role names were written on the board and students were asked to stick their user stories on the board, under the appropriate role. The class period ended with all students studying the board and attempting to group user stories into clusters of similar stories. At the end of class the user story clusters were collected and transcribed into a complete list that could be shared with the class.

All projects were too large for individual teams to handle so the next class session involved inspection of the combined user story list and discussion of how they could be parsed into between six and eight smaller projects. User stories were then assigned to each of the identified sub-projects with student teams selecting the part they wanted to work on. Once this was complete the teams began the first of three two-week sprints.

Each sprint followed the pattern described in Phase Two: Sprint planning meetings where prioritized items from the product backlog were selected and moved to the sprint backlog, execution of the sprint, sprint review and retrospective. As before, work that did not meet the definition of done was returned to the team's product backlog and no grades were assigned. In what is a deviation from Scrum designed to eliminate, or at least control for, the "free-rider" problem (The Nelson Touch Blog, 2011), individual sprint grades were weighted by results from a peer-evaluation. Finally, each team held a brief sprint retrospective and submitted a written summary to the instructor.

One fundamental component of Scrum that is missing from the preceding description is the daily Scrum, or daily stand-up meeting (Agile Alliance, 2015). Daily scrums are designed to have each team member answer three questions: 1) what did I accomplish yesterday, 2) what do I plan to accomplish today and 3) what is keeping me from reaching my goals?

The daily Scrum proved difficult to implement in the first attempt to add agility to the course for several reasons. First, the class doesn't meet daily and it isn't possible to control student team activity outside of class. Second, the class meets three times a week for 55 minutes. Given the limited amount of time available it was very difficult to allocate 15 minutes of each class to the meeting. Finally, our students generally communicate with each other and do a great deal of teamwork outside of class and didn't initially see value in "daily" meetings.

Our clients were working professionals in other parts of the university. As such it was difficult if not impossible to ask them to completely fulfill the role of product owner (Scrum Methodology, 2015). Most clients simply did not have the availability required for the job so the instructor filled in for them where possible. Student teams did, however, contact the clients via telephone, email and text to gather additional information and clarification of issues. Also, since most teams experienced difficulties reaching the definition of done, most actual client/product owner demonstrations were delayed beyond the scope of a single sprint.

Teams were encouraged to talk to each other during sprint planning and execution to assure integration and consistency. However, a scrum of scrums was not used during this first attempt and the majority of the integration occurred just prior to final client presentations.

Lessons Learned

Short classes make it very difficult to conduct in-class daily scrums (or stand-up meetings) and still allow time for teams to work together and to consult with the instructor, scrum master or product owner.

Returning unfinished work to the product backlog may work well with teams in the real world but it seems to promote loafing in the classroom environment.

Large, multipart projects happen in life but are extremely difficult to manage in the classroom. Smaller projects with independent teams appears to be a better approach for novice students.

COURSE STRUCTURE – SECOND YEAR

Phase One

During the second year phase one of the course was similar to the first year with readings, quizzes and discussion on agile methods and Scrum.

Student teams worked through a short series of training sprints but this time the focus was on the template web sites available in Visual Studio Express. Use of these templates allowed for the introduction of web sites created with Bootstrap (2016) style sheets and HTML.

While this phase didn't specifically migrate to the MVC (2016) framework, web pages were data-driven with both text and images dynamically uploaded and displayed.

Phase Two

Phase Two differed in two significant ways during the second year. First, we partnered with a local consulting company who both supplied the project and served as product owner to the teams. Second, we used a project that was small enough for all teams to work on independently and this was presented to the teams in the form of a challenge – who could develop the best solution, including a challenge to add gamification to the process.

The client provided an initial list of of product backlog items and also introduced several new tools to the class including Trello (2016), #slack (2016) at GitHub (2016). Trello boards were used to manage the product backlog and to make it visible to the team, the client and the instructor. Slack was used for communication between student teams and their product owners and github was used to manage and share source code between teams and the product owners.

The project consisted of four two-week sprints. Each sprint began with a planning session (graded on the basis of what was on the Trello board) and ended with a review and retrospective. Because the product owners were off-site with busy "real-world" schedules the reviews were done primarily by the instructor who compared the sprint backlog to the working, potentially releasable product. Unfinished work was returned to the product backlog and, as opposed to what happened the previous year, teams were graded based on what they had accomplished. Also, in an effort to minimize social loafing, team grades were weighted by peer evaluations to determine individual student grades.

The client returned to the classroom for the final product review in the form of a demonstration by each team. All teams were present for all demonstrations.

DISCUSSION, OUTCOMES AND BENEFITS

No quantitative data was collected during this introduction of agility to the capstone course. One project, a Career Marketing system to help students collect personal information and market themselves, is in trial use by the college at this time. This system implements many of the ideas that the Director of Career Planning had been trying to implement for the last 8-10 years. A second project attempted to simplify the review and approval of study abroad proposals. While not complete, the class developed a number of good insights and approaches that will, hopefully, lead to implementation of the system in the future. The final system, a human resource allocation system for our college of Osteopathic Medicine, is still in a rough prototype stage but the clients expressed a great deal of surprise at how much an undergraduate class was able to accomplish in a short amount of time.

Students on the second year project teams incorporated the client's color schemes into their web pages, figured out how to create and edit employee profiles, how to allow employees to recognize each other for their accomplishments and how to record and use that information in leader boards and badges of different types.

From an instructional point of view we determined that strict adherence to Scrum principles is difficult in an academic environment with 55 minutes classes meeting three times a week. Team members frequently worked individually and outside of class so there were problems with the cohesiveness of the development team, including, but not limited to the free-riders typically found in university team projects. While we were able to partially control this with the use of peer evaluations the problem didn't completely go away. Because of the time available, how frequently team members communicated with each other and how the teams worked on project tasks many teams were reluctant to spend class time on a daily standup meeting. We also experienced some problems with the responsiveness of our clients, both on-campus and off-site. Finally, all students were taking other classes, most with group projects so it was impossible for the scrum master to isolate them from outside pressures.

The Agile Manifesto (2001) lists four guiding values and has been supplemented by twelve principles (Agile Alliance, 2015B). One way to summarize the outcomes of this attempt to add agility to the classroom is to compare the guiding values and principles of agile with what happened in the course. Table 1 lists the values with comments on how they relate to the course design.

Table 1: Agile Values

Agile Value	Relation to Class
Individuals and interactions over process and tools	Teams are self-selecting and self-assigning. They follow the process as outlined here but do not use traditional approaches such as the Rational Unified Process, or tools such as process modeling and use cases.
Working software over comprehensive documentation	Product and sprint backlogs are produced along with summaries of sprint reviews and retrospectives, not long narrative documents with multiple signoffs
Customer collaboration over contract negotiation	Customers/product owners are involved in the entire process but no contracts related to what the final product will be
Responding to change over following a plan	Changes are welcome and generate new or revised backlog items that are processed like any other backlog items.

Table 2 lists some of the twelve agile principles associated with the Agile Manifesto and how they related to the class.

Table 2: Agile Principles

Agile Principle	Relation to Class
Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.	Attempt to have working software at the end of each sprint, not just at the end of the semester.
Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.	Product backlogs updated as needed. No more Microsoft project or updates on when we are on the plan.
Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for the shorter timescale.	See first point.
Business people and developers must work together daily throughout the project.	Frequent interaction between the development team and the product owner (a weak point the first year).
Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.	Student teams are provided with the tools and resources. Software that isn't "done" at the end of a sprint goes back on the product backlog without penalty.
The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.	Somewhat weak during the first year byt tempered by student preference for social media but favored over written documents
Working software is the primary measure of progress.	No more meaningless milestones
At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.	Short retrospective meetings at the end of each sprint

STUDENT REACTIONS

Student feedback was gathered from the standard course and instructor evaluation administered by the college and a separate questionnaire students were asked to complete. While no formal analysis was performed on these results, student feedback proved valuable in examining the success of our implementation efforts.

Students in one spring class listed some of the course strengths as "forced to gain experience when given non-specific requirements in order to produce deliverables", "the course is modern, team affiliated and works with valuable, applicable processes and information Working with SCRUM was excellent" and "working with teams in a project which is more applicable to the real world". They also complained that the grades weren't up-to-date (due to the process of sending incomplete work back to the product backlog). Some comments are ultimately related to the uncertainties of live client project work and the fact that Scrum is an iterative approach to project developments as indicated in the values and principles listed above: "the instructor didn't always explain what needed to be done", and a need to "create clear modules of what needs to be accomplished in the time frame".

The second spring class had a somewhat easier project and their comments indicated that the "Course is relevant to what we will be doing in the real world" and "Helped us learn a new process hands-on (SCRUM)" but that the instructor "could plan for more time at the end for the client project" and "provide easier way to collaborate besides 5 people on one computer".

In the separate questionnaire one student commented:

"The most important things I learned revolved around communication amongst team members, proper sprint planning, proper sprint execution, and not being afraid to ask team members for help on certain aspects of the project.

The course was different from what I expected because there was less direction than I thought. We were given a lot more freedom and flexibility to carry out projects in our own way.

These were also reasons I thought the course was valuable because you can learn from mistakes more easily when you do things your way and you're the one that's accountable."

When asked what aspects of Scrum development work best, one student in the second year class offered the following comments:

"Principle 'Accept that you can't get it right up front' because we had a lot of issues and it was a learning process

I enjoyed following the principle 'to keep options open until the last responsible moment', because we were constantly adapting when some things worked and others didn't when we had different members working on different things."

Another student in the same class described the things that worked best by restating major features of Scrum:

"Having a goal for each sprint

Planning before each sprint what we want to accomplish

Reflecting on sprints to understand what we accomplished and what we need to do next

Asking each other what we can do differently next sprint to improve on our last sprint"

CONCLUSIONS AND FUTURE DIRECTIONS

Overall student feedback on the approach taken is quite positive but indicates areas that need attention. Quite a bit of feedback was related to the quizzes at the beginning of the semester. Based on the feedback provided more time needs to be spent discussing the content of the readings and videos. A number of students mentioned a need for more time on the client project and that can be arranged by reducing the programming review, the number of new tools introduced and the number of training sprints.

There were also a significant number of comments related to the structure of projects and the amount of detail provided on assignments. Some of these comments are related to the design of a prerequisite course where assignments typically use a much more structured, step-by-step approach than what happened in this course. Other aspects of this problem can be addressed by spending more time on the underlying meaning of the core values and principles of Scrum and what they mean for project work.

Some students commented that the Scrum framework broke down when they were working on the client project. Some of this can be addressed by insisting on Stand Up sessions in each class once the project has begun, stricter penalties for missing class work sessions and increased efforts to show how different aspects of project work relate to Scrum.

We attempted to address additional issues raised during the first year with better-defined projects with more supervision during the second year. We worked with several alumni at a national consulting company who wanted to partner with our program and they assisted us in selecting an appropriate project, coaching and serving as product owners. The main problems we encountered with this approach were related to the client being seventy miles away and the creation of less than ideal communication paths between students and their client product owners.

Some of the pedagogical conclusions from the first year resulted in changes to the approach used during the second year. Among these was the need to quickly grade and provide feedback on sprint planning, sprint reviews and sprint

reflections. While this appears to go against the spirit of Scrum it does appear to be necessary to keep the class on track.

In a similar fashion sprint teams should be self-organizing and in control of their own activities while teams of students require more supervision, more reasons to be in class and need feedback from their peers. Grading on attendance and using peer evaluations to determine individual grades can help achieve those ends.

REFERENCES

#slack (2016) https://trello.com (last accessed 27 March 2016)

Agile Alliance (2015). Daily Meeting, http://guide.agilealliance.org/guide/daily.html (last accessed 28 May 2015).

Agile Alliance (2015b). The Twelve Principles of Agile Software, http://www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software/ (last accessed 28 May 2015)

Agile Modeling (2015): User Stories: An Agile Introduction, http://www.agilemodeling.com/artifacts/userStory.htm (last accessed 28 May 2015).

Baird, A., & Riggins, F. J. (2012). Planning and sprinting: Use of a hybrid project management methodology within a CIS capstone course. *Journal of Information Systems Education*, *23*(3), 243.

Beck, K. & Andres, C. (2014) Extreme Programming Explained: Embrace Change, 2nd Ed. Addison-Wesley.

Bootstrap (2015) http://getbootstrap.com (last accessed 27 March 2016)

Cohn, Mike, "Differences Between Scrum and Extreme Programming", https://www.mountaingoatsoftware.com/blog/differences-between-scrum-and-extreme-programming (last accessed 25 June, 2016)

Deved˘zic, V., & Milenkovic, S. A. R. (2011). Teaching agile software development: A case study. *Education, IEEE Transactions on*, *54*(2), 273-278.

GitHub (2016) (last accessed 27 March 2016)

http://agilemanifesto.org, (2001). "Manifesto for Agile Software Development". *(accessed 27 May, 2015)*

Lu, B., & DeClue, T. (2011). Teaching agile methodology in a software engineering capstone course. *Journal of Computing Sciences in Colleges*, *26*(5), 293-299.

Luce, T. (2014). "Using Standards and Best Practices to Help Students With Limited Technical Skills Create Full-Featured Web Sites" in *Issues in Information Systems*, 15(1), 209-216.

Luce, T. & Matta, V.(2010). Academic Calendar Change Induced Curriculum Change in *Issues in Information Systems*, 11(2), 145-151.

Mahnic, V. (2010). Teaching Scrum through team-project work: students' perceptions and teacher's observations. *International Journal of Engineering Education*, *26*(1), 96

MVC (2016) http://www.w3schools.com/aspnet/mvc_intro.asp (last accessed 27 March 2016)

Rubin, K. S. (2012). *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Signature Series.

Schwaber K. & Beedle. M. (2001) Agile Software Development with Scrum, Pearson.

Scrum Alliance (2014). Key Elements in the Sprint Retrospective, https://www.scrumalliance.org/community/articles/2014/april/key-elements-of-sprint-retrospective (accessed 28 May 2015)

Scrum Alliance (2015). https://www.scrumalliance.org (accessed 28 May 2015)

Scrum Alliance (2016). https://www.scrumalliance.org/why-scrum (accessed 28 June 2016).

Scrum Methodology (2015) Scrum Product Owner, http://scrummethodology.com/scrum-product-owner/ (last accessed 28 May 2015)

Scrum Training Series (2015). http://scrumtrainingseries.com. (last accessed 28 May 2015).

Scrum.org (2015). https://www.scrum.org (accessed 20 May 2015)

Soundararajan, S., Chigani, A., & Arthur, J. D. (2012, February). Understanding the tenets of agile software engineering: lecturing, exploration and critical thinking. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 313-318). ACM.

Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time*, Crown Business.

Tan, C. H., Tan, W. K., & Teo, H. H. (2008, April). Training students to be agile information systems developers: A pedagogical approach. In *Proceedings of the 2008 ACM SIGMIS CPR conference on Computer personnel doctoral consortium and research* (pp. 88-96). ACM.

The Nelson Touch Blog (2011) Team Incentives and the Free Rider Problem, https://nelsontouchconsulting.wordpress.com/2011/03/17/team-incentives-and-the-free-rider-problem/ (last accessed 28 May 2015).

Trello (2016) https://trello.com (last accessed 27 March 2016)

VersionOne (2015). https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf (accessed 6 April, 2016) *10th"* Annual State of Agile Report".

YouTube (2015). NEW Intro to Agile Scrum in Under 10 Minutes – What is Scrum? https://www.youtube.com/watch?v=XU0llRltyFM and SCRUM01-SCRUM10 https://www.youtube.com/watch?v=ACcfYB_5iV8&list=PLebXUHgaZIX59v5YBgxRYCV7batgaCN2G&index=1 (last accessed 28 May 2015)