Journal of Information Systems Education

Volume 31
Issue 3
Summer 2020

An Agile Framework for Teaching with Scrum in the IT Project Management Classroom

Daniel E. Rush and Amy J. Connolly

Recommended Citation: Rush, D. E. & Connolly, A. J. (2020). An Agile Framework for Teaching with Scrum in the IT Project Management Classroom. *Journal of Information Systems Education,* 31(3), 196-207.

Article Link: http://jise.org/Volume31/n3/JISEv31n3p196.html

Initial Submission:	30 May 2019
Accepted:	17 October 2019
Abstract Posted Online:	4 June 2020
Published:	8 September 2020

ISSN: 2574-3872 (Online) 1055-3096 (Print)

An Agile Framework for Teaching with Scrum in the IT Project Management Classroom

Daniel E. Rush
Information Technology and Supply Chain Management
Boise State University
Boise, ID 83725, USA
danrush@boisestate.edu

Amy J. Connolly
Computer Information Systems and Business Analytics
James Madison University
Harrisonburg, VA 22807, USA
conno3aj@jmu.edu

ABSTRACT

This paper presents a framework for teaching a complete, semester-long IT project management course with traditional PMI-based content (sans software development) while featuring Scrum as the organizing logic for accomplishing coursework. This framework adapts widely-used Scrum practices from industry for use in the classroom, including how to organize student teams, homework, and activities. Organizing an existing course with Scrum is intended to maximize student learning of traditional project management content, as well as the difficult-to-teach, socially-complex, "soft" skills that lead to Scrum team success. This deep integration of Scrum into a traditional, predictive IT project management course goes well beyond single activities or units without crowding out valuable time and material. A brief overview of the agile philosophy and examples of teaching Scrum in the classroom situate this work in the teaching and learning literature. Classroom-tested Scrum rituals and example artifacts are provided to illustrate how to apply the framework. This group-based, iterative, and hands-on approach equips students to better internalize and understand the complex social interactions involved with a self-organizing team, concepts that are difficult to learn without first-hand experience. The proposed framework will help IS educators implement Scrum practices in their own courses, further addressing industry's increasing demand for IS professionals with Scrum experience.

Keywords: Project management, Agile, Scrum, Pedagogy, Teaching framework, Active learning

1. INTRODUCTION

This paper presents a teaching framework for integrating Scrum in a traditional, predictive IT project management course. The goals of this framework are threefold: (1) to teach modern agile principles to upper division students with varying degrees of project experience *independently* from software development, (2) to teach traditional project management techniques and tools as embodied in the Project Management Institute's (PMI) Project Management Body of Knowledge (PMBOK), and (3) for self-organizing student teams to develop important soft skills (e.g., peer leadership, conflict resolution, and communication). Agile is arguably the most prevalent philosophy for quickly and responsively developing software, and agile frameworks such as Scrum and eXtreme Programming have gained significant adoption in software development curricula (Devedzic and Milenkovic, 2011; Mahnic, 2012; Lang, 2017). However, scant literature explores how to implement agile frameworks to teach project management to IS business students. In 2015, as the primary author was preparing a project management course, he could not find examples of a course designed to meet all three goals. Since then, a small but growing body of pedagogical research has emerged with examples of teaching agile concepts in IS. However, none of these examples yet provides a tangible framework for teaching a whole course in project management. As a result, this framework was built out of necessity and is shared here for other IS educators looking for a tested approach.

The current literature suggests that when IS students learn about agile concepts, it is in an introductory or limited fashion – as one topic among many, instead of internalizing the full agile process. For example, students receive introductory activities or lessons that teach the "what" of agile without practicing the "how" and "when" (e.g. Saade and Shah, 2016; Sibona, Pourreza, and Hill, 2018). While an introduction is necessary and useful, Scrum involves much more, and one or

two exercises on agile are unlikely to provide students with sufficient group interaction to develop the social acumen needed in industry (Baham, 2019).

Agile focuses on individuals, collaboration, working output, and adaptive response to change (Beck et al., 2001), meaning that soft skills (communication, collaboration, and flexible adaptation) are key to successful agile teams. Agile teams rely on communication to succeed (Hummel, Rosenkranz, and Holten, 2015). However, project management curricula has traditionally struggled to teach soft skills (Pant and Baroudi, 2008; Clarke, 2010) despite employers' continued demand for talent to communicate, adapt, and work effectively in project teams. A deeper engagement with agile experiences in the IT project management classroom will help prepare IS students to meet this persistent demand for soft skills.

To best realize the benefits of an agile framework such as Scrum, students need to internalize the method in order to gain confidence in their ability to use it in future work environments. One way to practice and internalize concepts is through active learning. Active learning approaches have been shown to effectively increase student performance (Freeman et al., 2014), and variations of active learning have long been present in the project management classroom in one form or another (Allan, 1999). Repetition and practice are hallmarks of active learning approaches because they emphasize deep learning, understanding, and accountability (Lipman, 2003; Warburton, 2003; Richmond, Boysen, and Gurung, 2016). We used an active learning approach to build the framework for teaching this course.

This framework provided students with a breadth of scenarios via classroom-based experiences. From these experiences, students developed confidence and demonstrable agile skills through repeated hands-on practice with Scrum rituals and artifacts. Learning through repeated practice is the method advocated by industry experts to develop agile skills, whether through exposure to numerous case studies (Schwaber, 2004) or through intense corporate training with group-based, iterative planning exercises (Griffiths, 2005). Therefore, our method helps to further align IT project management curricula with industry best practice.

This paper describes the results of converting a 15-week, traditional IT project management course to a semester-long Scrum project schedule with 2- to 3-week Sprints using Scrum roles, rituals, and artifacts to restructure the coverage of the course's original project management content. Rather than bolting-on a single lesson, activity, or module about the agile philosophy or one of its associated frameworks, this hands-on Scrum approach gives students months of experience developing their expertise in Scrum artifacts and the socially-complex rituals of Scrum, and in doing so, better prepares them for agile projects.

We anticipate this framework should be of use to anyone who teaches or studies project management, especially today, as the field begins navigating how to complement training in traditional predictive approaches (e.g., SDLC and waterfall) with frequently-used agile approaches, without sacrificing content. The agile philosophy grew out of software development. Therefore, one challenge in implementing agile in the IT project management course is the lack of software or systems development; students plan to build a project but they typically do not actually build one. Our motivating question is: How can a college course in predictive IT project management be restructured without a software development component so that students develop the skills to confidently work in agile Scrum teams?

This work contributes to the literature in information systems, IT project management, active learning pedagogy, and beyond. To explain how we designed the framework and provide valuable background context, we first review selected literature on agile and Scrum practices, focusing on the relationship of agile to traditional project management. We then describe how we structured a semester-long course to practice agile project management principles while still teaching traditional project management content. We describe the use of hands-on exercises implementing the principles of agile and show how the in-class exercises directly reflect industry practice. We present preliminary lessons learned from teaching the course twice, as well as student reflections. We conclude with reflections and possibilities for future research.

2. LITERATURE REVIEW

2.1 Agile in Teaching and Learning

The importance of incorporating the agile philosophy into project management curriculum is noted in the literature (Bredillet et al., 2013), however research on how to do this for IS and business students is still emerging. This is in direct contrast to teaching agile frameworks to software developers, of which there are many examples. For instance, Lang (2017) proposed "agile learning" in the context of a web app development course. Other software development examples include McAvoy and Sammon (2005), Devedzic and Milenkovic (2011), and Mahnic (2012). However, without software to develop, how does one provide opportunities for students to practice agile concepts on a project?

Recently, a handful of IS pedagogy articles have shown how to teach independent Scrum exercises in systems analysis and design (May, York, and Lending, 2016) and the *Journal of Information Systems Education* published a special issue dedicated to teaching agile in IS. This issue explored methods to teach either "What" agile is or "How" it can be implemented in a non-agile classroom (Sharp and Lang, 2018). Even so, the majority of the *JISE* special issue focused on programming or systems analysis and design, courses which are more related to software development rather than project management (Chen and Rea, 2018; Linden, 2018; Magana, Seah, and Thomas, 2018; Taipalus, Seppänen, and Pirhonen, 2018). The *JISE* special issue included two papers with examples of using agile in project management or a general MIS course (rather than software development), but even these had only one or two broad lessons but no examples of how to teach agile or Scrum in the rest of the course (Schmitz, 2018; Sibona, Pourreza, and Hill, 2018). In order to teach students the socially-complex nature of agile teams, teams must be given sufficient time to norm and perform, processes that take time (Tuckman and Jensen, 1977).

Other than two recent conference presentations (Javadi and Tanner, 2018; Owens and Shekhar, 2018), the most similar work we could identify to this one reported on a Master's level agile project management class in which the entire course was converted to a Scrum-like, Sprint-based structure (Cubric, 2013). In that course, the class replaced software development with wiki edits and recorded what they learned about agile

project management in a collaboratively-produced artifact. The results indicated that increased communication and Sprint planning contributed to group cohesion and that teamwork, negotiation, and mutual respect were all significantly enhanced. The framework presented in the present paper provides similar team results, but in an undergraduate course and with deliverables and team activities that students are more likely to encounter in a general industry setting (e.g., project charters, Gantt charts, earned value spreadsheets, and team presentations) as opposed to Wiki pages. To the best of our knowledge, our paper is among the first to present a framework for teaching Scrum all semester in the IT project management course, without having students design software. Although we believe there are examples of IS instructors using agile for project management, these are not readily available for use.

2.2 What is Agile Project Management?

The agile philosophy encompasses any method that supports the *Agile Manifesto*'s values of "Individuals and interactions over processes and tools, Working software over comprehensive documentation, Customer collaboration over contract negotiation, and Responding to change over following a plan" (Beck, et al., 2001). Agile frameworks dominate software development and include methods such as Crystal, Dynamic Software Development Method, feature-driven development, Lean software development, Scrum, Extreme programming (XP, XP2), and variations such as "Scrumban" which combine Scrum and Kanban techniques (Dybå and Dingsøyr, 2008; Stettina and Hörz, 2015). Since the Manifesto was declared by software developers in 2001, this philosophy has expanded to other areas of business, such as service delivery (Kowalkowski et al., 2012), general business processes (Graml, Bracht, and Spies, 2008), and business intelligence (Larson and Chang, 2016).

The widespread adoption of these techniques and their importance to project management are reinforced in the latest revision of the PMI's PMBOK Guide (2017a), which for the first time came bundled with a companion book, "The Agile Practice Guide" (2017c) recognizing agile's growing place in project management. In project management, agile is sometimes contrasted with predictive waterfall methods; however, they are not orthogonal. In fact, companies can and do manage projects in a hybrid manner, mixing these methods, depending on the size, type, and needs of the project (West et al., 2011). Multiple examples of mixed-methods approaches from companies such as Caterpillar, Blue Cross Blue Shield of Nebraska, IBM, and City Furniture are relayed in the PMI's Pulse of the Profession report, "Achieving Greater Agility" (PMI, 2017b). The theme of the report emphasizes that delivery approaches are selected based on the organization's needs and project characteristics, that PMO functions such as minimizing risk and controlling costs are still required for agile projects, and that project managers should be adaptable and well versed in agile and predictive approaches.

2.3 Scrum Methodology

Scrum is one of the most popular of the agile methods and is a process model of project management because it specifies a process that is performed iteratively until a business owner declares the output complete. Scrum relies on small teams of practitioners who self-govern and organize themselves using the processes prescribed by Scrum (Schwaber, 2004). One of Scrum's central tenets – and the hardest to learn – is how to effectively operate in a self-organizing team (Kropp et al., 2014). Thus, Scrum instruction that aspires beyond the introductory level should challenge students to engage deeply and repeatedly in a variety of team-based interactions that occur while practicing Scrum rituals. These interactions guide students to develop experience and social competence in self-organizing. As Kropp et al. (2014) discuss, the agile competencies most in need of being developed go beyond engineering principles and management practices. From a learning theory standpoint, these social competencies are on a higher level than technical skills alone.

Effective student learning of these skills occurs when students "[develop] and [discuss] agile values and attitudes" (Kropp et al. 2014, p. 144), which is achieved when students gain "personal experience … socially through realistic discourse" (Kropp et al., 2014, p. 143). We organized our instruction to develop this socially-complex set of competencies by creating opportunities for students to engage in realistic discourse. We believe this structure is recognizable as Scrum while striking a balance between giving students enough freedom to find their own answers and a clear enough structure to feel confident in their learning. We next describe the classroom site and how Scrum was integrated into the curricula, both in general terms and with concrete examples.

3. COURSE SETTING

3.1 Classroom Site

This teaching framework was implemented across two years of an upper-division IT Project Management ("ITPM") class at a mid-size teaching university in the western United States. Offered each spring, the class serves as an elective for business-focused computer information systems (CIS) students and as a required course for software engineering (SE) students. As a result of this compositional mix, students begin the course either familiar with software development but with minimal experience in business fundamentals (e.g., time value of money, people management, and presentation skills) or vice-versa. Regardless of background, the course provides students with the opportunity to learn the breadth of project management concepts and their connections to business fundamentals, independent of software development.

3.2 Supporting Textbook and Projects

The project management course content before and after modification was based on the PMI's *PMBOK Guide, 5th ed* (2013). For the two semesters described, Kathy Schwalbe's *IT Project Management, 8th edition* (2016) was adopted, which is structured generally by PMI Knowledge Areas (KA) and Process Groups (PG). The course also included three of Schwalbe's "running cases" which are simulated IT implementation projects. For example, the cases cover projects wherein students organize a global entrepreneurial event or estimate the costs of implementing energy-efficient hardware upgrades in various companies. These cases include specific deliverables and activities aligned with each of the PMI KA's and PG's. Team deliverables from these cases were project artifacts such as a business case, project management plan, Gantt chart, etc. Running cases are commonly used in IT project management to simulate the concepts of completing a project (Austin, Nolan, and O'Donnell, 2009). In addition to team

Sprint	M/W/F, 6-Sprint Topic Areas	T/Th, 5-Sprint Topic Areas
Pre-Sprint (PS)	How class works, Agile (SCRUM) in the classroom, Intro to PM, MS Project	How class works, Agile (SCRUM) in the classroom, Intro to PM
Sprint 1 (S1)	PM in the IT Context, MS Project, PM Process Groups	PM in the IT Context, MS Project, PM Process Groups, Project Integration Management
Exam 1 (E1)	One class-period exam	One class-period exam
Sprint 2 (S2)	Project Integration Management, Project Scope Management	Project Scope Management, Project Time Management (1st half)
Sprint 3 (S3)	Project Time Management, Project Cost Management, Facilitated Sprint 3 Retrospective in Session 24	Project Time Management (2nd half), Project Cost Management, Project Stakeholder Management, Project Communications Management
Exam 2 (E2R, E2)	Review one day, Exam the next	One class-period exam
Sprint 4 (S4)	Project Quality Management, Project Human Resource Management	Project Human Resource Management, Project Quality Management
Sprint 5 (S5)	Project Communications Management, Project Risk Management	Project Risk Management, Project Procurement Management
Sprint 6 (S6)	Project Procurement Management, Project Stakeholder Management	N/A
Final Exam (E3)	Extended time-period exam	Review in Session 30, Extended time-period exam during finals week

Table 1. Topic and Sprint Schedules for 15-Week Semesters

deliverables, students completed individual quizzes and exams and a number of individual assignments. The approximate distribution of work was 60% individual (15% participation, 15% quizzes, and 30% exams) and 40% team (Sprint homework). A list of topics covered in the course is presented by Sprint for both 2- and 3-day-a-week meeting schedules in Table 1.

In the pilot semester, the course met 3 days a week on Monday/Wednesday/Friday for 50-minute class periods. In the second iteration, the course met on a Tuesday/Thursday schedule for 75-minute class periods which affected the number of in-class Scrum meetings that were possible. The more frequent meetings of the 3-day-a-week schedule felt more Sprint-like, but the Sprint review sessions were less compressed in the 2-day-a-week schedule due to an extra 25 minutes per class session.

4. COURSE STRUCTURE

4.1 Non-Scrum Classroom Time

One of the benefits to this framework is that the Scrum activities described in the following sections complement rather than monopolize instructional contact time. Much of the time dedicated to Scrum activities is spent discussing or reviewing traditional project management content in teams or as a class, and after all activities are accounted for, approximately 60% of the contact hours remain unaffected and available for non-Scrum instruction (such as lectures), activities, and individual assessment. While Sprint planning and Sprint review meetings take up much (or all) of the class meetings in which they occur, during the remaining class sessions the Daily Scrum meeting was the only time dedicated to Scrum. After the Scrum meeting concluded, the remaining time was spent with mini-lectures, discussions of book topics, exercises to practice concepts, etc. We found the time for these activities sufficient to engage with all the traditional project management content covered by a course that did not adopt the Scrum structure.

4.2 Semester Schedule

The course was taught during a 15-week semester with a one-week, mid-semester break. In addition to the team-based work on the running case studies and other Schwalbe homework, two mid-semester exams and a final exam were administered after the first, third, and last Sprint. These exams and an associated review session were outside of (between) Sprints and were designed to assess individual understanding of the PMBOK material.

4.3 Scrum Training and Team Formation

The semester started with a short pre-Sprint period in which students learned the agile philosophy and Scrum concepts. The use of Scrum teams was introduced in the first class, with an assigned reading from Schwalbe and the Scrum Reference Card (James, 2012). The readings were followed by a lecture on Scrum, with example Scrum videos and artifacts shown in class and available on the course Learning Management System (LMS). The introduction to Scrum in the pre-Sprint period was similar to the individual activities or lessons recently reported in the literature (e.g., May, York, and Lending, 2016; Sibona, Pourreza, and Hill, 2018). What differentiates our framework from these excellent introductions is the repeated practice of Scrum rituals throughout the course, rather than moving on to non-agile topics after the agile lectures and activities. This new method provided teams with more opportunities to develop and practice self-governance and other critical "soft" skills. Again, what's new here is the opportunity for students to "close the loop" to see how the entire Scrum process works and to make changes to how they execute the process over the length of the semester.

The instructor assigned teams based on students' availability to meet outside of class, along with major, year-in-program, and other considerations to ensure a mix of experience

and ability on each team. In the second semester, students were asked to participate in a collective decision making process to develop and adopt a set of rules for group discussions. These rules were brainstormed in the first class meeting and a subset was adopted in the next class meeting.

4.4 Five 2-to-3 Week Sprints

The 15-week semester was divided into approximately 2-week Scrum Sprints (6 in the pilot semester and 5 in the second semester due to days of instruction (TR versus MWF)), with an initial one-to-two weeks for Scrum training and team formation plus time for semester exams and final exam review. During each Sprint, students chose, committed to, worked on, and presented the results of homework assignments from the book or a running case. Each Sprint lasted four to six class sessions.

We next describe the details of Sprint structures and how we adapted industry practice to a classroom setting. We then supplement this description with concrete examples from Sprint 5 in the second iteration of the class to show how it worked. Additionally, the idealized industry experience and what was happening in the classroom were emphasized to students as part of their learning in the first few Sprints, to make these experiences seem more realistic.

5. USING THE SCRUM METHOD TO TEACH IT PROJECT MANAGEMENT

5.1 Scrum Description

In the ITPM classroom, we implemented a version of Scrum containing widely used roles, standardized meetings ('rituals'), and artifacts, examples of which are shown in Table 2, adapted from Schwaber (2004). These core Scrum roles and rituals prepare students for organizations with customized Scrum implementations (e.g., Business Analyst or other areas of expertise instead of a generic "Team Member" role).

Scrum meetings scaffold selecting work, monitoring progress, confirming acceptance of the final deliverables, and improving the work processes. In the ITPM classroom, this repetition provided a structure and cadence for each Sprint. Students internalized this pattern through repeated practice over the semester, allowing their knowledge of Scrum to move from "what it is" to "how we produce work with it."

Scrum Roles	Product Owner, ScrumMaster, Team Member
Scrum Rituals	Sprint Planning Meeting, Daily Scrum Meeting, Sprint Review Meeting, Sprint Retrospective Meeting
Scrum Artifact Examples	Product Backlog, Sprint Backlog, Task Board, Sprint Burndown Chart

Table 2. Sprint Roles, Rituals, and Example Artifacts

A figure of a typical Sprint iteration is shown in Figure 1 (adapted from James 2012). To work in an educational setting, this general pattern was adapted from industry and supported with Scrum artifacts and rituals, which we instantiated in the classroom. A description of how each Scrum component might be generally used in an industry setting can be readily found in materials such as the PMI publication "The Agile Practice Guide" (2017c) or Schwaber's book *Agile Project Management with Scrum* (2004). We adapted these rituals to the classroom, as explained in the next section.

5.2 Scrum Applied in the Classroom

Adapting Scrum to the classroom consisted of altering pre-semester preparation activities and adapting the in-class schedule to accommodate Scrum activities. The following sections describe how each Sprint ritual or artifact was adapted for the classroom, after which concrete examples drawn from a single Sprint in the second iteration are shown.

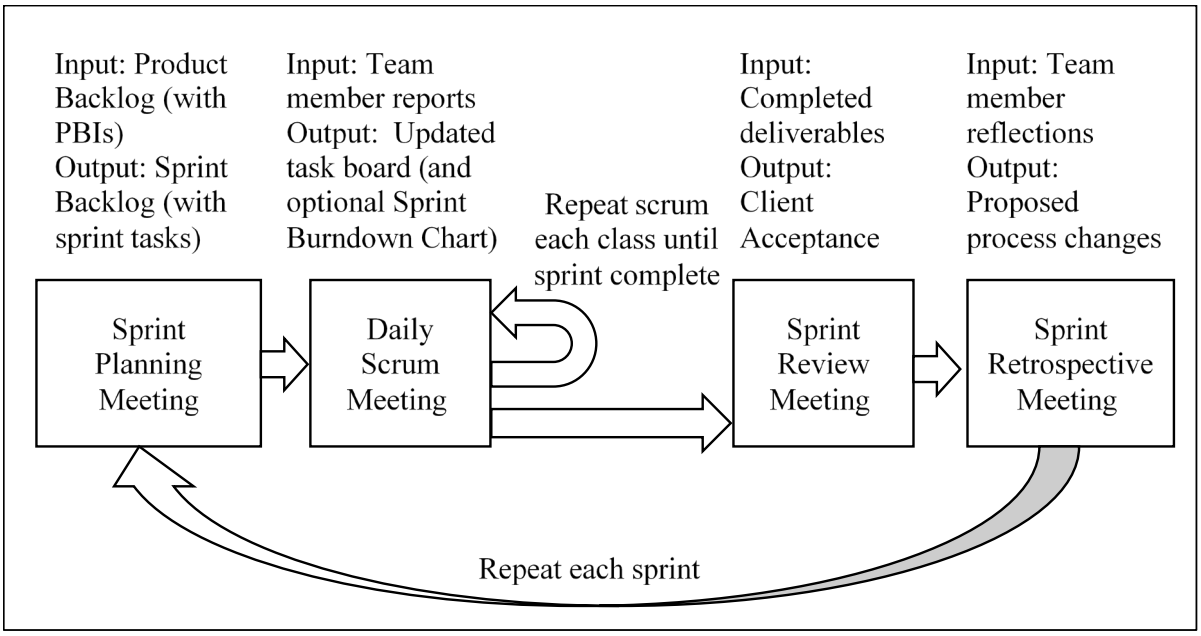


Figure 1. Typical Sprint Process (adapted from James, 2012)

5.2.1 Preparation prior to Sprint. Prior to assigning work to the Scrum teams, the instructor first developed the assignments and organized them into backlog items. In contrast to the user stories common in industry, classroom PBIs were assignments from the book or tasks associated with a running case. These assignments covered the KAs and PGs taught near the Sprint in which the tasks were assigned. These tasks showcased how the instructor adapted industry practices for the classroom.

5.2.2 ScrumMaster and Sprint planning meeting. At the beginning of each Sprint, the student teams elected a ScrumMaster. Each team member was required to serve as ScrumMaster at least once during the semester, to give every student an opportunity to practice that role's responsibilities: coordinating the team's efforts, resolving blocking issues, and reporting on the progress of deliverables. After selecting a ScrumMaster, the team held a Sprint planning meeting with the product owner (the instructor) to review and prioritize PBIs in the product backlog. Attendance at this meeting was mandatory. The team selected how many PBIs they estimated they could complete during the Sprint. The team was then committed to these PBIs for the duration of the Sprint. Finally, as the last part of the Sprint planning meeting, the team divided each PBI into Sprint tasks, estimated the amount of effort required to complete the tasks, and assigned each task to a responsible person. These tasks formed the committed Sprint backlog (the output of the Sprint planning meeting). The ScrumMaster then represented the team's backlog in a visually accessible manner on a Kanban board (an example is provided in section 5.3.2) for the team to reference and update during daily Scrum meetings.

The instructor provided a force ranked list wherein each PBI was given an importance priority (from 1 to 8, with 1 being the highest) and a point value (from 5 to 25 in 5-point increments). The point values were not equivalent to function points used in software development complexity, but they served as a guide for students to estimate the amount of work per deliverable. The point values represented the maximum possible grade for a successfully completed deliverable. Students could elect as a team to attempt any number of PBIs, although the maximum total points for each Sprint was capped at a specific total points possible. In theory, a team that did not attempt enough PBIs might earn less than 100% homework grade, but generally teams chose enough PBIs for the possible point total to meet or exceed the cap (the latter to earn the maximum allowable credit for the Sprint, even if they did not earn full points on all the PBIs).

The student teams used the in-class planning meeting to select PBIs and obtain instructor clarification on expectations, akin to a project meeting with a customer. During these meetings, the instructor clarified course concepts associated with PBI deliverables such as the Gantt chart, Pareto chart, stakeholder response strategy, etc. Each team worked its own running case, with no more than two teams using the same case. The decision to use multiple cases gave students the opportunity to see examples of deliverables applied to multiple scenarios.

5.2.3 'Daily' Scrum meetings. In the classroom, similar to industry daily Scrum meetings, the ScrumMaster lead 5- to 15-minute meetings at the beginning of each class period. The ScrumMaster asked each team member to report: (1) What have I (the member) completed since the last meeting (and what work do I have remaining)? and (2) What issues are blocking progress on my assigned items? The ScrumMaster recorded each member's report on the task board. Starting with the Sprint that covered the time management KA, the remaining work estimates from the daily reports could optionally be aggregated in a Sprint burndown chart (as one of the graded PBIs) to visualize work completed over the length of the Sprint.

Kanban task boards were used to facilitate transparent and open communication. In the beginning of the semester, teams could use either individual white boards or a large pad of poster paper for their task board. They then represented Sprint backlog items and tasks on sticky notes placed in the column that corresponded with their current state (e.g., Committed, Not Started, In Progress, or Completed). An example Kanban board is provided in the Example Sprint section. As the semester progressed, teams were encouraged to try building their Kanban boards in electronic collaboration tools such as Trello. Even after experimentation for a Sprint or two, most of the teams (~80% each semester) returned to using physical boards.

5.2.4 Sprint review meeting. In the classroom, the Sprint Review was held during class time at the end of each Sprint. Each team presented the end status for each of the PBIs they attempted in the Sprint Backlog. Teams earned full points for those items that the instructor (acting as the Product Owner) accepted, reduced points for items provisionally accepted, or zero points for items not accepted. During the next Sprint, teams could optionally choose to reattempt provisionally accepted items and were required to reattempt items not accepted. If accepted on the next Sprint, teams received full points for reattempted items. While slightly more formal than a functional review around a developer's workstation, the public presentation of work offered repeated practice in communication skills. As each running case was assigned to at least two teams, this structure ensured at least one team in the audience was an expert on the presented running case and could ask informed questions. While presentation professionalism was not explicitly graded, students reported feeling accountable to their peers for communication skills. An unintended outcome of these presentations is that some students reported a deeper understanding of the assigned artifacts and/or tasks after seeing how they were applied to multiple running case examples.

5.2.5 Sprint retrospective meeting. Similar to industry but unlike previous implementations of Scrum in the classroom, each team held a Sprint retrospective meeting after each Sprint to reflect on their work processes during the Sprint and to identify successful elements to be repeated or improvements to be made. In preparation for the retrospective, students completed a set of questions for the instructor which asked students to reflect on their personal experiences. After composing responses and sharing them privately with the instructor, students discussed their answers in their Sprint teams and agreed on any tool or process changes to implement in the next Sprint.

As an opportunity to reflect on how work was done, the retrospective plays an important part in the Scrum process by giving teams a space to discuss and work through friction points, such as responsiveness on different communication channels or dealing with free riders. To help ameliorate free-

Sprint 5 Product Backlog

Max points to be earned this sprint: 40

Sprint 5 Items (from Chapters 8, 11, & 12)

Priority	Ch.	Description	Points
1	8	List of quality standards/requirements. Global Treps Running Case Tasks 1 & 2 (p. 339) / Green Computing Part 5 Task 1 + running case task 2 from book (p. 339) / Manage your health Part 5 Tasks 1 & 2	10
3	8	Exercise 6 (p. 337) – Lean quality assurance and using Kanban cards	10
6	8	Pareto chart (Global Treps running case task 3 (p. 339) / Green Computing – use exercise 2 on p. 337) / Manage your health Part 5, Task 3	5
9	8	Exercise 4. (p. 337) – Research Malcolm Baldrige National Quality Award	5
4	11	Risk Register and probability/impact matrix (Running case tasks 1 & 2; Part 8 for Green Computing and Manage your health; p. 461-462 for Global Treps)	10
7	11	Response strategy (Running case task 3; Part 8 for Green Computing and Manage your health; p. 462 for Global Treps)	5
5	12	Running Case #3 – Proposal Evaluation Spreadsheet (Health: Part 9 #2, Green IT: Lessons learned report)	5
8	12	Exercise #3 (p. 491) – Lease vs. Buy Analysis	5
2	12	Exercise #4 (p. 491) – IT contract analysis (type, key clauses)	10

** Super Tool

Sprint 5 Management Notes:

A formalized sprint backlog (indicating tasks, statuses and time estimates) should be generated for every class period and the set is to be turned in at the end of the sprint. These are included in the ScrumMaster's homework grade.

A sprint burndown chart may be generated for an additional 5 points (up to the 40 point maximum).

Figure 2. Example Sprint Product Backlog

riding, the instructor solicited confidential concerns via the individual written feedback that each student was required to turn in associated with a retrospective, as well as by offering to mediate any team issues that appeared to be at an impasse. During the two semesters reported on in this paper, no teams utilized this service, and qualitative results indicated that while free-riding did occur, teams were able to handle it on their own, further evidence that students internalized the Scrum concept of self-organizing teams. The next Sprint began with a kickoff meeting (and a new product backlog from which to pull items) during the next class meeting.

5.3 Example Sprint

Next we present Scrum rituals and artifacts from Sprint Five of the second course iteration.

5.3.1 Preparation for the Sprint and product backlog. Prior to the Sprint 5 Planning Meeting, the instructor prepared a Product Backlog for the Sprint by populating it with Product Backlog Items (PBIs) drawn from or related to the following chapters in the course textbook: chapter 8 "Project Quality Management," chapter 11 "Project Risk Management," and chapter 12 "Project Procurement Management." The PBIs have two primary sources: end-of-chapter exercises related to the knowledge area (e.g. "Research the Malcom Baldridge award" is an exercise from chapter 8) and the "running cases" from Schwalbe. Each simulated "running case" has relevant exercises and deliverables associated with each chapter (e.g., creating a list of quality standards is running case task 1 for the "Global Treps" project). The three cases are titled "Global Treps," "Green Computing," and "Manage your Health." The "Global Treps" case is in the printed 8th edition of the textbook and the other two cases are available on the first author's website. After selection, each PBI was assigned a prioritization (analogous to a product owner's indication of importance) and an associated point value (analogous to the amount of work in 'planning points'), as shown in Figure 2.

5.3.2 Sprint planning meeting, ScrumMaster, and Kanban task board. At the start of each Sprint, a 20- to 30-minute planning meeting was held. During the Sprint planning meeting, each team elected a ScrumMaster, reviewed the Product Backlog, and selected which PBIs they wanted to attempt for the Sprint. In the Sprint 5 example, 9 prioritized PBIs (plus the always available Sprint burndown chart) show as available, and each PBI is worth 5 to 10 points. While the 10 potential PBIs total 70 points together, the maximum points each team could earn for the Sprint was capped at 40 points. This cap was set by the instructor to help the team commit to a realistic workload, while still exercising choice over their work. With a 40-point cap, teams typically selected PBIs worth 40 to 45 points. Each of the items was transferred to a Kanban task board (see Figure 3) for tracking and communicated to the

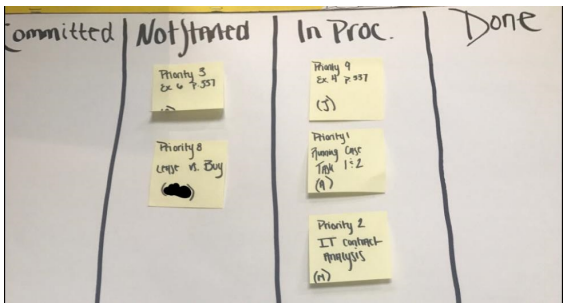


Figure 3. Example Kanban Task Board

Product Owner (the instructor) by the ScrumMaster. In any remaining time during the meeting, teams were encouraged (but not required) to break larger PBIs down into discrete tasks. For example, in the Kanban board shown in Figure 3, the PBI "IT Contract Analysis" (Priority 2, 10 pts) could have been broken down into 3 sub-tasks: "create framework with types and typical clauses," "analyze contract," and "write report with results." However, the team elected to keep this PBI (and all other PBIs) as one task each. After determining tasks for the PBIs, the teams collaboratively assigned who would work on which tasks and estimated how long each task was expected to take.

5.3.3 Daily Scrum meeting. Frequency of Scrum meetings varied by team, with many holding them only during class, despite being informed of the opportunity to hold additional Scrum meetings on their own. During the Scrum meeting, as each team member answered the two questions, the ScrumMaster recorded each member's report in a log, updated the task board with the current status, and worked to resolve blocking issues (either during the remaining meeting time or afterwards with the instructor and others). In the example shown, three tasks are shown as in progress and two are not yet started. The remaining work estimate from the daily reports could then be aggregated in a Sprint burndown chart to visualize work completed over the length of the Sprint (this was required in one Sprint, and available as an optional five-point PBI in the remaining Sprints).

5.3.4 Sprint review meeting and deliverable acceptance form. A majority of time in the last class period of each Sprint was devoted to the Sprint review meeting. During the Sprint review meeting, each team presented their completed work on the selected PBIs to the Product Owner (the instructor) and the class. Presentations were typically supported by slides (in PowerPoint or Google slides) and sometimes included live demonstrations, such as if the PBI was to construct or populate a project management tool (e.g., a Gantt chart with dependencies). Teams answered questions from the class and the product owner, who then chose to accept (either in full or with partial points), not accept, or defer the acceptance decision on each PBI. The deferral option was used when the quality of a particular deliverable was not apparent from the presentation or the deliverable was a short paper that needed to be read prior to acceptance (e.g., PBI Priority 8, Lease vs. Buy analysis). After evaluating all PBIs, the product owner returned a copy of the "Deliverable Acceptance Form" to the team's ScrumMaster.

If any items were accepted but did not earn full points or if they were not accepted, they could be reattempted for full credit in the following Sprint. In the example deliverable acceptance form shown in Figure 4 (recreated from an actual form to mask identities), the PBI "Lean Quality Assurance" (labeled "Exercise 6, p. 337" on the Kanban board, Priority 3, 10 pts) was accepted, earning the full 10 points, while the PBI "IT Contract Analysis" (Priority 2, 10 pts) was at first deferred, and then upon review only earned 9.5 of the 10 possible points, making it eligible for a reattempt if the team desired to do so in the next Sprint.

5.3.5 Sprint retrospective meeting. Due to limited class time, the Sprint Retrospective meeting was held in class for only the first Sprint, and teams were responsible for holding retrospectives outside of class for all subsequent Sprints, with a written summary turned in to the instructor. During the first, facilitated retrospective, the instructor introduced and collected responses to a set of questions, including items such as "What tools do you want to collaborate with in the next Sprint?" and "What went well and what could be improved? (Write down 2-3 of each)." These prompts were re-sent electronically after each Sprint Review and teams were encouraged to meet and discuss them.

6. STUDENT REFLECTION AND DISCUSSION

6.1 Student Reflection on Scrum Mastery

Student reactions to this novel class structure were evident in their course evaluations in Spring 2016. More detailed responses were solicited through an informal survey in Spring 2017. Students reported their experience and comfort with Scrum and agile methods before and after the course. The use of both sources was approved by the university's institutional review board. Students in the second iteration of the course reported a high degree of self-confidence in their ability to apply Scrum techniques to future projects, despite only two students having used Scrum before and one-third having never heard of Scrum or agile methods before taking the class. Of the two students who had heard of or used Scrum, one indicated learning about Scrum and using it in the workplace and another indicated using Scrum to develop software in one or more other classes. The student with industry experience wrote,

> I currently work at a tech start-up, where all of our processes are derived from SCRUM methods. I had never heard of SCRUM before working at the company. I was extremely pleased to see it finally introduced in my studies since it is widely used in the IT industry, specifically in software engineering.

In contrast, the student who was familiar with Scrum but hadn't used it wrote,

> I had heard of SCRUM and Agile methods and was supposed to learn them in-depth in the Systems Analysis and Design class, but [we] barely scratched the surface on how the method actually works. We were not given any assignments on the topic or tested on the material.

At the end of the semester, all students agreed or strongly agreed that they knew the Sprint structure and the four primary Scrum rituals by the end of the class. They also expressed a high degree of confidence (Median: Strongly Agree) in their ability to act as ScrumMaster and create a Kanban board and "Agreed" that they were comfortable creating a burndown chart or status report for future Scrum projects. Students also Strongly Agreed (Median: 5, Mean: 4.6) with the statement "I would recommend that future Project Management classes adopt a similar SCRUM team and Sprint structure for their homework."

This preliminary feedback suggests that repeated exposure to Scrum rituals and artifacts equipped students with the knowledge, tools, and confidence to use Scrum techniques in

Sprint 5
Product Owner (Client) Acceptance Form

Project Name: Green IT
Team Number: 1
Team Members: Alice Student, Latisha Student, Javier Student, Mei Lee Student

ScrumMaster: Mei Lee Student
Point Summaries: Current Sprint Max: 40,
Current Sprint Attempted: 40

Present Order	(Deliverable Priority #) and Description	Max Points	Accepted? (Product Owner Only)
1	Priority 1: Running Case Task 1 & 2	10	Yes- 10
2	Priority 2: IT Contract Analysis	10	? 9.5
5	Priority 3: Lean Quality Assurance	10	Yes- 10
3	Priority 8: Lease VS Buy	5	? 4
4	Priority 9: Malcolm Baldridge National Quality Award	5	Yes- 5
	Prior Sprint Backlog Items	N/A	

- - - - - - - - - - - - - - - - - - To be filled out by the product owner - - - - - - - - - - - - - - - - - -

I (We), the undersigned, acknowledge and accept delivery of the work indicated as completed for this project on behalf of my (our) organization. My (Our) signature(s) attest to my (our) agreement that the listed deliverables have been completed. No further work should be done on these deliverables.

Name	Title	Signature	Date
Dr. Professor	Product Owner / Assistant Professor	Dr. Prof	5/1/17

1. **Was this project completed to your satisfaction?** **Yes** **No**
2. **Please provide the main reasons for your satisfaction or dissatisfaction with this project.** Well done overall, Quality assurance and Quality award deliverables are exceptionally strong!
3. **Please provide suggestions on how our organization could improve its project delivery capability in the future.** See comments on Contract analysis and Lease vs Buy deliverables for suggestions to strengthen. These may be re-attempted for full points next sprint (if desired)

Figure 4. Deliverable Acceptance Form (Black text indicates "During Sprint Review" grading and red text indicates post-review comments)

future projects. In written responses, the most common theme was that the Scrum teams and Sprint structure made the class and homework more enjoyable, efficient, and effective. For example, one student wrote,

> The Scrum and Sprint structure really nailed how Scrum and agile project management can be applied to projects because we, the students, were using it ourselves to complete actual projects. It was very hands-on and a good/fun way to teach Scrum and agile project management.

Students also anticipated workplace benefits from learning Scrum. Comments included "The real world application of Scrum was very fascinating and helpful. This should be continued."

6.2 Impact of Scrum on PMBOK learning

At first, the instructor was concerned that because teams were allowed to (1) select a subset of possible homework deliverables and (2) divide up the work on those deliverables in whatever way they desired, not all students would engage with and learn how to create all of the assigned PMI-recommended tools (e.g., project charter and Gantt chart). This concern was mitigated by three strategies: (1) by identifying and assigning high point values to "super tools" so that teams would select them, (2) by assessing individual knowledge of course concepts, techniques, and tools through quizzes and exams, and (3) by requiring the whole class to be present for all Sprint Reviews, which allowed each student to see one or more applications of each deliverable on two or three different running cases. One student commented that "Requiring groups to present their information helps a lot with having to get to know what you're working on better, rather than just writing an assignment down and forgetting about it instantly after."

6.3 Soft Skills Acquisition through Active Learning and Reflection – The Self-Organizing Teams

While this course taught traditional project management concepts, it was structured based on the principles of active learning, which meant students created meaning through personal experience. In active learning, after an exercise or practice session, students reflect on their learning to instill mindfulness, codify what they've learned, and commit that learning to memory. Students complete this necessary, final step rather than the instructor summarizing the day's material. Retrospection is an important step in active learning as well as in Scrum, but in Scrum, retrospection helps the team improve in future iterations. In a Scrum team, members reflect on their work at multiple points. During the Sprint, each daily Scrum meeting is an opportunity to discuss what work is complete, what isn't, and what is impeding progress. At the end of the Sprint, students presented their completed work to their peers. Finally, the Retrospective meeting at the end of each Sprint served as a formal mechanism for each student to reflect on and discuss their individual performance in the Sprint – good and bad – as well as how the team worked together to identify what they wanted to change to improve their process in the next Sprint.

A common theme in students' written reflections was the accountability and authority within the self-organizing teams. Six of the 10 comments that addressed this theme spoke positively of the structure, and 3 suggested improvements. A student wrote "I really liked that Scrum made all team-members accountable for part of the group work. It was very easy to pinpoint if somebody was not pulling their weight and made it easy to address the issue if it came up" while another commented that there

> needs to be some way to give the team greater authority to hold social loafers more accountable, while eventually our teams [sic] social loafer did actually contribute something meaningful it took them all semester after many minor confrontations about the issue to finally contribute meaningful work.

One student suggested that "... the ScrumMaster [should] have a bigger say in the participation status of the group. In this way people who care about their grade will put more stock into what their ScrumMaster is asking of them." These responses suggest that effective self-governance was not achieved on their first attempt, but that students persisted in their refinements, and even at the end of the semester, were considering ways to improve the experience. This feedback further supports the benefits of a semester-long framework as opposed to one or two in-class exercises on Scrum.

7. CONCLUSION AND CONTRIBUTION

This paper reports on a Scrum-based approach to teach traditional project management content in an undergraduate IT project management course. Through repeated practice, students developed feelings of competence in socially-complex soft skills, which are one of the most difficult concepts of agile-inspired project management approaches, yet can have outsized impact on project success. This framework was implemented twice: first in a class of 15 students, then in a class of 30. Rooted in self-organizing teams, this framework makes many other innovative changes possible, and faculty are encouraged to adopt, modify, and improve the framework presented in this paper. For example, faculty choosing to emphasize work breakdown structures and scheduling could populate the product backlog presented to the students with assignments drawn from the PMI Practice Standard for Work Breakdown Structures (PMI, 2019b) or the Practice Standard for Scheduling (PMI, 2019a).

In reflecting on our own use of this framework, the authors have identified several adaptations to adopt in future iterations of the course. One adaptation appropriate for courses in which students work on a variety of real projects (not case-based) is to expand the list of PBIs available to choose from to include items suited to meet the particular needs of the projects the students are working on (e.g., user stories for a training handbook). Another adaptation would be to support the retrospective process with software to systematize the gathering of feedback and give the faculty member the ability to monitor that teams are utilizing the process even when it occurs outside of class.

Finally, we have noticed that students in classes that meet three times a week appear to feel more comfortable with Scrum earlier in the semester than students in classes that meet two or fewer times a week. A possible idea to address this might be to specify a minimum number of Scrum meetings per week. Other faculty are invited to adopt and adapt the framework to other schedules and delivery modalities (e.g., quarter system or hybrid-delivery).

Due to the limitations inherent in the size of the program where this approach was applied, several possibilities for furthering this research would require the participation of the larger ITPM-teaching faculty community. One opportunity available in settings with multiple sections of ITPM would be to teach ITPM using a traditional approach in one section and with the described framework in a second section, and then compare student knowledge of agile and soft skills between the two. For faculty who have the opportunity to sequence one or more development classes after the ITPM class, one possibility for future research would be to teach the described Scrum concepts in ITPM and then apply them in a later software development class. Such an approach would allow for data to be gathered on student knowledge and comfort with Scrum at each stage of the learning and application process, possibly

providing insight into the best manner in which to deploy the described framework.

One of the strengths of this framework is that the Scrum structure overlays existing course schedules, lectures, and homework without crowding out fundamental content. This structure allows the course to support the rich learning that occurs in socially-complex Scrum teams without sacrificing material. Although agile practices are taught in software development, based on our review of the literature, agile frameworks have yet to be fully embraced in the IT project management curricula over the entire course. As a result, although industry sorely needs agile talent, students are not developing the confidence and social skills they need to succeed on agile teams until well after graduation. Therefore, IT project management instructors may be interested in this framework for adapting the ITPM course to include a fully agile project. Agile methods are in high demand, and this paper presents one way project management curricula can adapt in order to prepare our students for their agile future.

8. ACKNOWLEDGEMENTS

The authors would like to thank several early readers and presentation attendees for their comments on this project, including Jeffrey May, Robert Anson, audience members at ICIS 2018, and the COBE research brown bag luncheon attendees. We would also like to thank the editors and anonymous reviewers who contributed to the development of this paper, including the suggestions regarding re-titling the deliverable acceptance form, the inclusion of the PMI's Practice Standard guides, and future research to assess students' subsequent software development effectiveness after taking project management taught using the described agile framework.

9. REFERENCES

Allan, G. (1999). Getting Students to Learn about Information Systems Project Management: An Experiment in Student-Centered Learning. *Research in Post-Compulsory Education*, 4(1), 59–74.

Austin, R., Nolan, R., & O'Donnell, S. (2009). A "Novel" Approach to the Design of an IS Management Course. *Communications of the Association for Information Systems*, 24(1), 315-332.

Baham, C. (2019). Implementing Scrum Wholesale in the Classroom. *Journal of Information Systems Education*, 30(30), 141-159.

Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Agile Manifesto*. Retrieved August 6, 2020, from http://agilemanifesto.org.

Bredillet, C. N., Conboy, K., Davidson, P., & Walker, D. (2013). The Getting of Wisdom: The Future of PM University Education in Australia. *International Journal of Project Management*, 31(8), 1072–1088.

Chen, K. & Rea, A. (2018). Do Pair Programming Approaches Transcend Coding? Measuring Agile Attitudes in Diverse Information Systems Courses. *Journal of Information Systems Education*, 29(2), 53-64.

Clarke, N. (2010). Projects are Emotional: How Project Managers' Emotional Awareness can Influence Decisions and Behaviours in Projects. *International Journal of Managing Projects in Business*, 3(4), 604–624.

Cubric, M. (2013). An Agile Method for Teaching Agile in Business Schools. *The International Journal of Management Education*, 11(3), 119–131.

Devedzic, V. & Milenkovic, S. R. (2011). Teaching Agile Software Development: A Case Study. *IEEE Transactions on Education*, 54(2), 273–278.

Dybå, T. & Dingsøyr, T. (2008). Empirical Studies of Agile Software Development: A Systematic Review. *Information and Software Technology*, 50(9), 833–859.

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active Learning Increases Student Performance in Science, Engineering, and Mathematics. *Proceedings of the National Academy of Sciences*, 111(23), 8410–8415.

Graml, T., Bracht, R., & Spies, M. (2008). Patterns of Business Rules to Enable Agile Business Processes. *Enterprise Information Systems*, 2(4), 385–402.

Griffiths, M. (2005). Teaching Agile Project Management to the PMI. In *Agile Development Conference (ADC'05)*, 318–322.

Hummel, M., Rosenkranz, C., & Holten, R. (2015). The Role of Social Agile Practices for Direct and Indirect Communication in Information Systems Development Teams. *Communications of the Association for Information Systems*, 36(1), 273-300.

James, M. (2012). *Scrum Reference Card*. Retrieved January 30, 2017, from http://scrumreferencecard.com/.

Javadi, E. & Tanner, S. (2018). Design and Implementation of an Agile Teaching Framework. In *Proceedings of the 24th Americas Conference on Information Systems*, New Orleans, Louisiana.

Kowalkowski, C., Kindström, D., Alejandro, T. B., Brege, S., & Biggemann, S. (2012). Service Infusion as Agile Incrementalism in Action. *Journal of Business Research*, 65(6), 765–772.

Kropp, M., Meier, A., Mateescu, M., & Zahn, C. (2014). Teaching and Learning Agile Collaboration. *Presented at the Software Engineering Education and Training (CSEEandT)*, 2014 IEEE 27th Conference (pp. 139–148), IEEE.

Lang, G. (2017). Agile Learning: Sprinting Through the Semester. *Information Systems Education Journal*, 15(3), 14-21.

Larson, D. & Chang, V. (2016). A Review and Future Direction of Agile, Business Intelligence, Analytics and Data Science. *International Journal of Information Management*, 36(5), 700–710.

Linden, T. (2018). Scrum-Based Learning Environment: Fostering Self-Regulated Learning. *Journal of Information Systems Education*, 29(2), 65-74.

Lipman, M. (2003). *Thinking in Education (Second Edition)*. Cambridge: Cambridge University Press.

Magana, A. J., Seah, Y. Y., & Thomas, P. (2018). Fostering Cooperative Learning with Scrum in a Semi-Capstone Systems Analysis and Design Course. *Journal of Information Systems Education*, 29(2), 75-91.

Mahnic, V. (2012). A Capstone Course on Agile Software Development Using Scrum. *IEEE Transactions on Education*, 55(1), 99–106.

May, J., York, J., & Lending, D. (2016). Play Ball: Bringing Scrum into the Classroom. *Journal of Information Systems Education*, 27(2), 87-92.

McAvoy, J. & Sammon, D. (2005). Agile Methodology Adoption Decisions: An Innovative Approach to Teaching and Learning. *Journal of Information Systems Education*, 16(4), 409–420.

Owens, D. & Shekhar, G. (2018). Using SCRUM Principles to Transform the Classroom. In *Proceedings of the 24th Americas Conference on Information Systems*, New Orleans, Louisiana.

Pant, I. & Baroudi, B. (2008). Project Management Education: The Human Skills Imperative. *International Journal of Project Management*, 26(2), 124–128.

PMI. (2013). *A Guide to the Project Management Body of Knowledge (PMBOK Guide), 5th ed.*. Newtown Square, Pennsylvania: PMI Publications.

PMI. (2017a). *A Guide to the Project Management Body of Knowledge (PMBOK Guide), 6th ed.* Newtown Square, Pennsylvania: PMI Publications.

PMI. (2017b). Achieving Greater Agility: The People and Process Drivers That Accelerate Result, *Pulse of the Profession In-Depth Report*. Newtown Square, Pennsylvania: Project Management Institute.

PMI. (2017c). *Agile Practice Guide*, *New Edition*. Newtown Square, Pennsylvania: Project Management Institute.

PMI. (2019a). *Practice Standard for Scheduling, 3rd ed.* Newtown Square, Pennsylvania: Project Management Institute.

PMI. (2019b). *Practice Standard for Work Breakdown Structures, 3rd ed.* Newtown Square, Pennsylvania: Project Management Institute.

Richmond, A. S., Boysen, G. A., & Gurung, R. A. R. (2016). *An Evidence-based Guide to College and University Teaching: Developing the Model Teacher*. Philadelphia, Pennsylvania: Routledge.

Saade, R. G. & Shah, S. (2016). Exploring an Agile Learning Activity to Teach Agile Project Management. In *Proceedings of Informing Science and IT Education Conference* (In SITE) (pp. 95–101). Vilnius, Lithuania: Informing Science Institute.

Schmitz, K. (2018). A Three Cohort Study of Role-Play Instruction for Agile Project Management. *Journal of Information Systems Education*, 29(2), 93-104.

Schwaber, K. (2004). *Agile Project Management with Scrum*. Redmond, Washington: Microsoft Press.

Schwalbe, K. (2016). *Information Technology Project Management, 8th ed*. Boston, Massachusetts: Cengage Learning.

Sharp, J. H. & Lang, G. (2018). Agile in Teaching and Learning: Conceptual Framework and Research Agenda. *Journal of Information Systems Education*, 29(2), 45-52.

Sibona, C., Pourreza, S., & Hill, S. (2018). Origami: An Active Learning Exercise for Scrum Project Management. *Journal of Information Systems Education*, 29(2), 105-116.

Stettina, C. J. & Hörz, J. (2015). Agile Portfolio Management: An Empirical Perspective on the Practice in Use. *International Journal of Project Management*, 33(1), 140–152.

Taipalus, T., Seppänen, V., & Pirhonen, M. (2018). Coping with Uncertainty in an Agile Systems Development Course. *Journal of Information Systems Education*, 29(2), 117-126.

Tuckman, B. W. & Jensen, M. A. C. (1977). Stages of Small-Group Development Revisited. *Group and Organization Management*, 2(4), 419–427.

Warburton, K. (2003). Deep Learning and Education for Sustainability. *International Journal of Sustainability in Higher Education*, 4(1), 44–56.

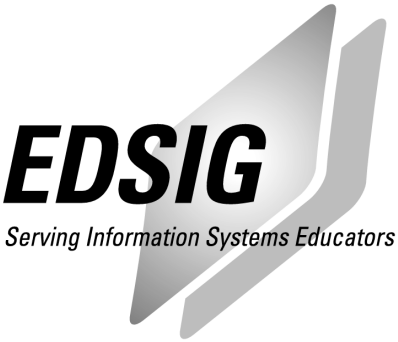
West, D., Gilpin, M., Grant, T., & Anderson, A. (2011). Water-Scrum-Fall is the Reality of Agile for Most Organizations Today. *Forrester Research*, 26, 1–15.

AUTHOR BIOGRAPHIES

Daniel E. Rush is an assistant professor of information technology management at Boise State University's College of Business and Economics. He earned his Ph.D. in business administration from the University of Michigan, where he studied business information technology and was part of the technology and operations department at the Stephen M. Ross School of Business. Prior to joining academia, he worked with IT projects in the health care, real estate, telecommunications, and high tech industries. Rush researches applying information systems to challenging interdisciplinary problems such as environmental sustainability, as well as topics related to project management and information systems education. His research has been published in *Journal of Cleaner Production*, *Communications of the Association for Information Systems*, and *Journal of Information Systems Education*.

Amy J. Connolly is an assistant professor of computer information systems and business analytics in the College of Business at James Madison University. Her doctorate is in management information systems from the University of South Florida. Her research interests include the role of social media in volunteer organizations and active learning and inclusion in information systems pedagogy. Her research has been published in journals including *European Journal of Information Systems*, *Information Systems Education Journal*, and *Informing Faculty*.

Information Systems & Computing
Academic Professionals



STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 2574-3872