

Introducing Agile Project Management Strategies in Technical and Professional Communication Courses

Journal of Business and Technical
Communication
2015, Vol. 29(1) 112-133
© The Author(s) 2014
Reprints and permission:
sagepub.com/journalsPermissions.nav
DOI: 10.1177/1050651914548456
jbt.c.sagepub.com



Rebecca Pope-Ruark¹

Abstract

Technical and professional communicators spend a good deal of time managing teams and documentation projects, and their organizations are increasingly introducing new project management practices. This article introduces Agile project management strategies that were created in software development environments, exploring how these iterative strategies can complement the traditional linear project management approaches that are taught in technical and professional communication (TPC) programs. To do so, the author presents a brief history of Agile, a case study of how the author applied specific Agile strategies in a grant writing course, and a comprehensive set of tips for implementing Agile in other TPC courses.

Keywords

curriculum, pedagogy, project management, Agile, collaboration

¹Elon University, Elon, NC, USA

Corresponding Author:

Rebecca Pope-Ruark, Elon University, Campus Box 2338, Elon, NC 27244, USA.
E-mail: rruark@elon.edu

Actively teaching collaborative project management strategies to technical and professional communication (TPC) students is crucial to their professional development because they will likely spend a good deal of their time managing projects in collaboration with other technical writers or in cross-functional teams. Our programs are responsive to these trends; in Allen and Benninghoff's (2004) survey of TPC programs in the United States, 30 of 42 programs reported that project management was either covered in all or most courses (11 programs) or featured in at least 1 or 2 courses (19 programs, p. 162). Collaboration and project management also ranked in the top nine topics emphasized in our programs (pp. 163, 165). In a more recent study, Meloncon and Henschel (2013) analyzed course catalogs for 65 TPC major programs and found that 24% of the programs required or offered electives on collaboration and 18% on project management (p. 50), and texts delving deeply into traditional documentation-management practices have served our programs well for years (Dicks, 2003, 2013; Hackos, 2006).

More recent trends in industry point toward the importance of being well versed in both the traditional, linear, plan-driven strategies and the new flipped, team-based, plan-emergent project management ideologies. Attention to Agile project management strategies, for example, has increased in recent years, with a number of book reviews about Agile practices appearing in *Technical Communication* and more presentations about technical writers in Agile work environments appearing at the yearly Society for Technical Communication (STC) Summits (e.g., Austin, 2012; Coker, 2010; Moore, 2013; Vaishampayan, 2012). The *Technical Communication Book of Knowledge (TCBOK)* that STC members are developing already has a page under a collaboration section exploring the benefits and challenges of Agile strategies for technical writers, yet the entire project management section of the *TCBOK* remains undeveloped, perhaps signifying that Agile is quickly growing in relevance to the field (Sigman, 2010). And according to technical writers working in Agile software development environments, writers have much more opportunity to advocate for users, express concerns and insights, and create more lightweight external documentation throughout short, iterative development cycles rather than focus on heavyweight internal documentation, ensuring better products and better supporting documents (Austin, 2012; Austin & Berry, 2011; Dayton & Barnum, 2009; Fox & Kramer, 2008). One technical writer to whom I spoke with at an Agile conference went so far as to advocate that those new to the profession should avoid becoming what he termed

“docudinosaurus” by becoming more versed in alternative project management practices such as Agile.

A few TPC studies have explored Agile in different environments, such as McNely, Gestwicki, Burke, and Gelm’s (2012) activity theory analysis of a student game development team using the Agile framework Scrum, Brown and Chao’s (2012) service-learning collaboration with a software development course, and my activities applying the Agile Scrum framework to full-semester service-learning projects in upper level TPC courses for majors (Pope-Ruark, 2012, 2014; Pope-Ruark, Eichel, Talbott, & Thornton, 2011), to name a few. By practicing both more traditional and new Agile strategies in the classroom, students may be better prepared for responsibilities that they will likely have in any future work environment, especially given the growing appreciation for Agile’s focus on self-organizing cross-functional teams, emergent planning, and iterative progress toward larger goals.

In this article, I introduce Agile project management as a set of methods that can complement traditional project management instruction in TPC courses. To explore this possibility, I first examine the context in which Agile was developed and then present a case study detailing Agile concepts and how I implemented Scrum, the most popular Agile framework, in an upper level service-learning TPC course in grant writing. Finally, I offer a set of specific tips for implementing Agile¹ and Scrum practices in collaborative projects of any length in TPC courses.

The Foundations of Agile

The term *Agile* describes a project management ideology articulated by 17 software developers in 2001. These “organizational anarchists” were responding to a critical and pervasive problem managing projects in software development by using linear “waterfall” strategies that divided development into discrete phases such as requirement gathering, planning, development, and testing (Highsmith, 2001). Because testing was relegated to the end of the process, and customers often changed their minds about what they wanted the software to do during development, the process was more often than not costly, off schedule, and frustrating (Cohn, 2010, p. 100; Freedman, 2009; Szalvay, 2004). Thus, developers who were “sympathetic to the need for an alternative to [extensive internal] documentation-driven, heavyweight software development processes” met and formulated the underlying ideology that would be Agile (Highsmith, 2001, ¶ 1).

What these self-named Agile Alliance developers came up with was a manifesto that lays out a set of values that recenter project management, moving away from what signatory Highsmith (2001) termed “Dilbert manifestations of make-work and arcane policies” to embrace practices that value people, customer input, flexibility, simplicity, reflection, and face-to-face communication (Beck et al., 2001; Beedle, 2006; Szalvay, 2004). *The Agile Manifesto* states that we should, whenever possible, value

individuals and interactions over processes and tools
working software over comprehensive documentation
customer collaboration over contract negotiation
responding to change over following a plan. (Beck et al., 2001)

Building on these foundations, a 2005 addendum called the “Declaration of Interdependence” added six core beliefs to the manifesto, essentially proclaiming the centrality of collaboration between team members, the team and the customer, and the team and the business representatives. These beliefs include bringing customers into the development process frequently, creating a work and team atmosphere that fosters both accountability and personal and professional growth and adapting fluidly to change by using situationally specific strategies, processes, and practices (Anderson et al., 2005).

ScrumAlliance (n.d.) reported multiple benefits for software development organizations that switch from traditional project management schemes to Agile, including faster development times with fewer flaws, and noted that superior Agile teams “achieve the Toyota effect: four times industry average productivity and twelve times better quality” (§ 3). When organizations achieve a high degree of Agile excellence, team members also tend to be happier, more engaged with their work, and more innovative in their approaches, thus producing more value for the organization and customers (§ 4; Grant, 2013).

Recognizing the need to prepare budding software engineers for these increasingly popular development environments, many academic programs have begun to include Agile in the curriculum, especially in client-based capstone courses. And in the early 2000s, articles published in computing science and engineering pedagogical journals began to cautiously explore Agile in the classroom, encouraging others to do so almost in the face of the inevitability of such practices in industry (Alfonso & Botia, 2005; Cleland, 2003; Hislop et al., 2002; Melnik & Maurer, 2003; Reichlmayr, 2003). More recently, researchers have found that students who complete Agile

projects make significant leaps as software developers regardless of their initial skill level (Perera, 2009), benefit (especially women and minorities) from the more social aspects of these learning environments (Slaten, Droujkova, Berenson, Williams, & Layman, 2005), and significantly improve their teamwork skills (Lingard & Barkataki, 2011).

Organizations implementing Agile certainly face challenges; the required shift in mind-set and management practice can be difficult to achieve across traditional organizations, and resistance to the intensive collaboration and self-organization that Agile involves is common because many developers are used to working alone and being told what to work on by a project manager (Dennings, 2012; Grant, 2013). Technical writers trained in traditional documentation management processes could face such challenges as well. Using these linear systems, they might suffer from the same issues as do software developers using waterfall practices. As technical writer Austin (2012) noted, many technical writers, like some developers, might prefer working alone rather than in teams.

Yet Agile's focus on process, self-organization, and collaboration has become increasingly important in industry and in academic programs training future professionals for these industries. Agile is not only popular in software development; a quick Google search reveals its reach in design, marketing, publishing, energy management, financial services, and civil and mechanical engineering, to name a few. Agile's emphases on people and teams, interactions with users throughout the process, and emergent and responsive documentation align with many of our arguments for humanistic elements of TPC courses and programs (C. Miller, 1979; T. Miller, 1991). For example, C. Miller (1989) later argued for an Aristotelian sense of the "practical" in that we characterize professional writing as a conduct rather than a production that values community and provides a "locus for questioning, for criticism, for distinguishing good practice from bad" (pp. 22, 23). The values and beliefs presented in the *Agile Manifesto* (Beck et al., 2001) and *Declaration of Interdependence* (Anderson et al., 2005) mirror Miller's assertions, moving software practitioners away from the overemphasis on product to a deep concern for a process that respects and values authentic communicative interactions between team members, business representatives, and customers in order to develop user-centered products, which is surely Aristotelian praxis. By teaching both more traditional and Agile project management, we offer our students multiple perspectives on the writing and development process, preparing them for a variety of work environments.

I have had promising success with using these approaches in many of my upper level service-learning TPC courses over the last 5 years (Pope-Ruark,

2012; Pope-Ruark et al., 2011). To explore how Agile might look in the TPC classroom and to introduce specific Agile concepts and strategies, I next offer a thick description of my fall 2012 course on grant writing for nonprofit organizations.

Scrum in a Course on Grant Writing for Nonprofit Organizations

I became interested in Agile and, in particular, Scrum, the most popular Agile framework, 6 years ago after being frustrated by the consistently mediocre collaborative projects I used in my business and technical writing courses (Pope-Ruark, 2012; Pope-Ruark et al., 2011). Students resisted collaboration and at best were able to cooperate for just a short period of time, typically dividing the work and putting it together into an inconsistent final document. Introduced to Scrum and its Agile strategies by my husband, a software engineer in Web environments, I began to slowly implement aspects of the Scrum framework into my client-based and service-learning courses with the goal of creating authentic and complex collaborative writing experiences. To grow my knowledge of Agile and Scrum, I participated in professional certification workshops for ScrumMasters (team facilitators) and Agile coaches, interviewed numerous Scrum professionals, and attended the international Scrum Alliance Agile 2013 conference, bringing that knowledge back to my classes.

Most recently I used Scrum to structure Grant Writing for Nonprofit Organizations, a special topics service-learning course offered in the Professional Writing and Rhetoric (PWR) concentration of the English major at Elon University. Designed primarily for PWR majors and minors, the course enrolled 18 students, and we partnered with five community agencies that offered services to survivors of domestic violence, previously incarcerated men, underprivileged children, older adults with memory or cognitive impairments, and exotic animals in need of sanctuary. To construct a context for the experience, I raised \$1,600 so that students and their community partners were vying for actual funds, adding urgency and authenticity to the course. Against this backdrop, I introduced students to the following Agile strategies to help them achieve course goals—sprints, Agile project terminology, Scrum folders, backlog grooming, daily Scrum, and reviews and retrospectives—each of which I detail within the context of my class.²

Sprints

In Scrum development environments, a sprint is a 15- to 30-day time box (i.e., time period) during which the teams commit to starting and finishing a set number of activities toward the larger project goals. Sprints essentially correlate with the scaffolded units used in many TPC courses. I used sprints to organize the course work so that the students and I could articulate how each activity informed the next piece of the larger project. Each sprint and its deliverable carefully contributed to the next sprint in necessary ways, allowing students to practice the skills that they would need to complete the next sprint as well as actively reflect on how their learning transferred to the next sprint. As such I created four sprints for the course:

- *Professional identity sprint.* For the first 2 weeks of the course, in what might be called Sprint 0, or the planning sprint, I introduced grant writing concepts and language, and I worked with students to articulate their professional identities. Given how crucial the presentation of an organization's ethos is in the grant writing process, this short warm-up sprint allowed students to explore their own professional ethos by creating personal profile documents to share with the community partners during the partner grant sprint. The sprint also required students to share their work regularly with peers, give and receive feedback, and meet deadlines, important skills necessary for the rest of the project.
- *Minigrant sprint.* To ensure students had one full grant writing experience early in the course, I had students then spend 2.5 weeks writing a short grant. I used an existing request for proposals (RFPs) from our College of Arts and Sciences because it required a relatively short narrative with explicit instructions laid out by the funding committee. During this sprint, we discussed the many issues of argument, arrangement, and style that are associated with a grant application and worked through them in their narrative drafts. Students went through the grant writing process from inception to submission and carefully reflected individually and as a group about what they learned and what they could improve for the next sprint, presaging the Agile planning and retrospective meetings that we would use in the next iteration.
- *Partner grant sprint.* During this 7-week sprint, groups of three and four students partnered with one community agency each to write a complete grant application. I developed the RFP, requiring a detailed

cover sheet, a 10-page narrative, and a complete budget. Because this sprint was longer than average, I added intermittent deadlines throughout to ensure that students received substantial feedback from peers, partners, and myself. For example, students worked with their partners during the first week to craft a one-page letter of inquiry confirming project direction and policies for communicating with each other. Interim deliverables allowed students to make continuous progress and add value to their final grants, receive consistent feedback, and experience team decision making, which is important in Agile development.

- *Funding-distribution sprint.* The final time box was a quick, 1-week sprint. After students completed their grant proposals, I informed them that they would determine the distribution of funds themselves and discussed how to use the previously available grant funding rubric to make their funding decisions. Thus, students were able to participate in the full spectrum of rhetorical activities related to grant writing and bring the project full circle.

At the class level, sprints were almost invisible to students because they were accustomed to scaffolded projects and lessons from other classes. Yet I used sprints to organize the way the course was run broadly and specifically, layering in other Agile concepts and practices as the sprints progressed in order to provide students with more project management and collaboration strategies to use.

Agile Project Terminology and Scrum Folders

Within sprints, Scrum practices encourage teams to break down project work into small slices that can be done by one or two people in a short period of time. In the case of the grant writing course, I introduced students to the concepts of epics, stories, and tasks that are frequently used by Scrum teams in order to formulate the slices of work that they take on during a sprint. Although Agile advocates might disagree on the exact definitions (Cohn, 2011), these concepts provide students a way to articulate the project in multiple levels of complexity:

- *Epic:* A complex software feature that is too big to complete in one standard 15- to 30-day sprint. For this class, I defined an epic as the primary sprint deliverable (i.e., the minigrant or the partner grant application) to help students see the product as the end goal but also

as something that required them to complete many other activities to be successful.

- *Story*: Something that a user wants to be able to accomplish with the software, such as to add page numbers in a word-processing application. I introduced stories as the smaller actions or documents that needed to be completed to finish the epic. In the partner grant sprint, a story might have been articulated as “draft budget narrative,” a piece of the grant proposal required to complete the epic.
- *Task*: Discrete slices of a story that must be completed for the story to work or be considered officially done. For the budget narrative story, one team listed tasks that included, among others, “contact partner to determine printer specs requested,” “conduct internet research to price printers,” “compare pricing,” and “choose printer for proposal.” Each research task was required to complete the story.

Visualizing the articulated stories and tasks on a Scrum board is important for both software teams and students because it encourages them to focus on the tasks at hand and allows them to see progress. While development teams might use a wall, a large whiteboard, or a software program designed to mimic a board, I implemented a more feasible folder system introduced by Benson and Barry (2011). In this system, each team received a manila folder and pack of 2-in. × 2-in. sticky notes. Using a marker and a ruler, each team divided the folder’s interior into three columns—“Backlog” (the Scrum term for work to be done over the course of the entire project), “WIP” (work in progress), and “Done” (see Figure 1). Students transferred each story and task onto individual sticky notes that they then prioritized on the Scrum folder. In my experience, students tend to articulate project activities at the story level, so providing teams with this language and visualization strategy seems to help them formulate discrete tasks that they can assign and complete. This use of sticky notes also allowed me to point out places on their folders where they were using a story as a task, helping them to further break down that activity and manage their work. We then discussed ways that they could use the folder to prioritize their work throughout the project and to guide each team meeting through a process called backlog grooming.

Backlog Grooming

A core Agile team activity called *backlog grooming* is the process in which team members articulate their project’s epics, stories, and tasks; estimate

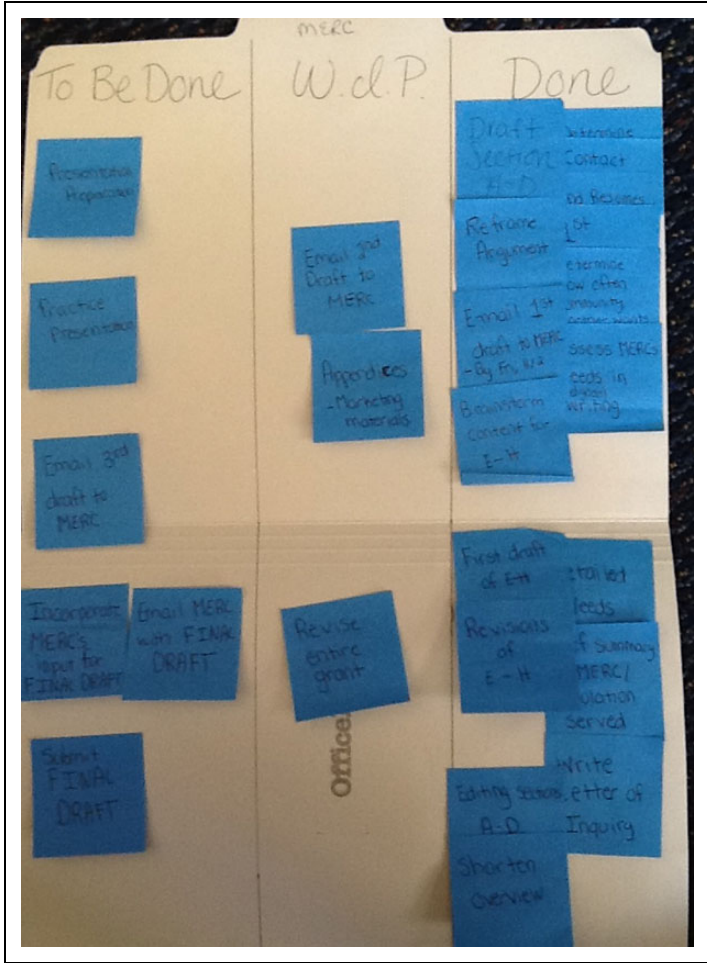


Figure 1. A sample student Scrum folder.

the time needed to complete each of these aspects of the project; and organize their priorities. The student teams were asked to start each team meeting by updating their folders, reprioritizing their tasks, and adding or removing any tasks from the backlog in response to changes as the project developed. Through this practice, the students could regularly reflect as a team on their collective activities.

For example, on the day the teams set up their folders, one team added sticky notes to their backlog column that read “write letter of inquiry,” “set up meeting with [partner],” and “write grant proposal draft.” At later meetings, students identified two of those items as stories (i.e., “write letter of inquiry” and “write grant proposal draft”) and then added discrete tasks to break each story down. Under “write grant proposal draft,” for instance, they added tasks such as “reread RFP and rubric,” “outline narrative section,” “ask [partner] for equipment model number,” and “research equipment costs.” They also added at the bottom of each task the initials of the student responsible for that task. Later in the project, teams also removed stories or tasks from the work flow visualized in their folders as they learned more about their projects. For example, one group had planned to ask for travel funding but removed associated tasks during a grooming session when they realized that the RFP specifically forbade it. Combining regular backlog grooming with the ritual of the daily Scrum commitment meetings, teams learned ways to stay on top of their project activities and their collaboration with each other.

Daily Scrum

At the start of the minigrant sprint, I introduced the practice of daily Scrum, which I referred to simply as Scrum, explaining that Agile teams meet every morning for 15 minutes so that each member can answer three questions: (a) What have I done since we last met? (b) What will I do now? and (c) What challenges or issues do I have that might benefit from the team’s input? (Schwaber & Sutherland, 2011, p. 15). More than a simple progress meeting, the daily Scrum allows team members to hold each other accountable to commitments and to ensure that no necessary task falls through the cracks, especially when others might be available to help.

The daily Scrum was a part of the class from the beginning. In the weeks leading up to the partner grant sprint, I convened an all-class Scrum once a week. Similar to a class discussion, students sat in a circle, and each student responded to the question at hand or offered possible solutions for the challenges that other students mentioned. We discussed, for example, their preconceptions about service and grant writing, skills and experiences that they might transfer to grant writing, and views on the minigrant both before and after the project was completed. By initially modeling the Scrum meeting this way, I hoped that students would become accustomed to speaking up and asking for assistance in the smaller groups. Once the teams were established for the third sprint, I encouraged students to continue Scrums in their

smaller groups along with their backlog grooming. The students then had the option of using the standard 15-minute Scrum meeting to kick off their work day and to refocus their attention on the tasks at hand for their grants. These daily group discussions paved the way for two different postmortem meetings at the end of the sprints.

Reviews and Retrospectives

Like planning and grooming meetings, reviews and retrospectives are an integral part of the accountability built into an Agile system such as Scrum. At the end of a sprint, the team participates in a *review*, during which they demonstrate the functionality they have built to stakeholders, managers, and other interested teams. The *retrospective*, on the other hand, is just for the team to discuss their processes, successes, and weaknesses during the sprint. Out of that meeting comes a small set of group-identified improvements that the team will commit to making during the next sprint, thereby showing their commitment to continuous improvement.

After the completion of each sprint, I asked students to reflect in two ways. First, I asked students to use the weekly journals that they kept throughout the course to reflect on their processes for that sprint, especially what they did well, what they would like to improve in the next sprint, and what skills or knowledge they had developed that could inform their activities in the next sprint. Second, I brought the students together so that they could verbally debrief the class on their process and learning. From my perspective, the daily Scrum, reviews, and retrospectives helped to normalize for the students the activities of self-reflecting, admitting to challenges, asking for help, and learning from successes and mistakes. These activities also allowed me to check in with the students to make sure they were following through on their goals for improvement during the next sprint and provide just-in-time teaching as necessary.

Like many Agile adoptions in organizations, Scrum adoption in this course was hit or miss (Grant, 2013), a recurring theme in all my Agile-based courses. In their final portfolio reflections, 6 of the 15 enrolled students who signed informed consent forms reported really liking the Scrum approach while four more appreciated it more at the beginning of the project. Teams that ultimately turned in the strongest grant proposals and felt that they had a strong collaborative environment used the Scrum folders extensively in their planning and early research and drafting activities. Once these groups overcame initial group-dynamic challenges and completed a full rough draft, they stopped using the folder but continued to use daily

Scrum so that team members would all be on the same page. One possible reason for this success was that these teams had a better handle on the requirements of the grant writing assignment and were able to integrate the Scrum activities into their process rather than see them as additional work.

Most of the teams reported using the Scrum folder and story–task distinctions actively during the first half of the project, but nine students reported that they abandoned or stopped using the folder as the deadline loomed. Students reported that they appreciated how articulating and visualizing activities in stories and tasks helped them to understand the work they needed to accomplish, set goals for progress, maintain accountability to each other, and celebrate small victories in progress. Toward the end of the project though, members of one team reported that they felt they had a good enough handle on the project and enough trust in each other to stop using the folder. Two other teams reported that the folder “just got away” from them later in the project and that they found it easier to use to-do lists and Google docs to manage the rest of the work. One student reported that the backlog column in the folder actually became more daunting and almost frightening as the deadline loomed and that a simple to-do list was more comforting.

Groups that chose not to use the folder were ultimately less successful, but this result is likely a symptom rather than a cause of group distress because these groups never overcame personality differences or trust issues in order to collaborate effectively. Two students reported actively disliking the Scrum process and saw it as not useful to their team (several members of less successful teams did not sign consent forms, so I could not include their insights here). A member of a team that fell apart noted in her final reflection that the folder and daily Scrum might have helped them stay on track but that they did not trust each other enough to try an unfamiliar process. Scrum and Agile are not magic bullets for improving collaboration between students or between professionals in industry. Scrum requires an ideological shift in the way that people view work and manage collaborations. But its tools and strategies can be useful in helping students work through some of the challenges associated with any type of collaboration, especially the types that they will encounter as TPC professionals.

In a survey conducted the following semester for a separate study, 91% of the students reported that the structure of the course helped them learn to collaborate more successfully whereas only 45% said the same about the structure of group projects in other classes. Based on these data and student feedback from reflections, I will continue to implement Agile practices in my courses to provide students with multiple project management strategies

and tools to draw on in future work environments. Despite the fact that many students became “less Agile” toward the end of the project, many experienced important gains in getting their projects off on the right foot and establishing a collaborative environment, which is important in any academic or professional TPC project. Certainly, the process is not immediately intuitive for students though many seem to appreciate the more practical strategies like the folder, daily Scrum, and story–task language. Because as professionals our students will encounter a variety of project types, team environments, and expectations for collaboration, introducing them to a variety of project management and collaboration strategies, including Agile systems such as Scrum, can add to their rhetorical toolkit in the workplace.

Tips for Implementing Agile Strategies

Whether you are interested in just dipping your toes into the pool or diving in completely, some forethought about the ways to introduce and adapt Agile to a TPC course context can go a long way. In this section, I offer specific tips on how to gain buy-in from students, use Agile vocabulary and visualization strategies in short projects, use more of the Scrum framework in longer projects, and determine whether or not using Agile is appropriate. Agile and Scrum strategies are not content dependent and can therefore be adapted to most TPC courses that use complex collaborative projects to foster learning, regardless of whether the students are, for example, TPC majors or engineering students in a required course. But regardless of project length, before introducing the project and Agile strategies to students, I highly recommend gaining buy-in from students early by discussing, validating, and using their own collaborative experiences to build a foundation for Agile teamwork:

- *Let students air dirty laundry.* On the first day of a group project, begin with an open discussion of the benefits and challenges of working with peers. Students are not often given class time to discuss and process their good and bad project experiences as learning opportunities. To start off the discussion, tell a few horror stories from your own experiences and encourage them to tell their worst team stories, making sure to have them summarize the worst characteristic of that experience (e.g., a social loafer, having differing expectations). Make a list of the worst characteristics on the whiteboard and have a note taker type it up for later reference.

- *Shift the emphasis.* Once students have created a list of bad team characteristics, ask them to describe their best team experiences and what made that group excellent. This list goes up alongside the bad team list so that you can compare effective group behaviors.
- *Create an expected-behaviors list.* Build on the two lists you have just created by encouraging students to pull out the four or five most important team behaviors that they will commit to enacting during their new group project. Post it somewhere visible so all teams can access it later, whether on a whiteboard or in a document on their course Web space.
- *Help students create a cooperative team environment themselves.* Once students are in their project teams, ask them to build on the class behavior list and to create their own team rules just like the new Agile teams do: How will they communicate with each other? What will they do if someone misses a meeting? How will they share knowledge? What is the process for resolving conflict? Going through these stages together can help students to develop what Rubin (2012) called “musketeer spirit,” that is, to see themselves as a team and to start building trust early on.

Once the stage has been set, you can introduce more specific Agile strategies to set up the project work. One of the draws of Agile is that the teams organize and direct their own work; students may not be used to such self-direction and may require some hand-holding until they begin to trust themselves and each other. For a shorter service-learning project that is essentially one long sprint, consider implementing the following basic strategies:

- *Have students articulate their project tasks.* Once the students have posted their team behavior lists to their team folder or shared online space, ask them to carefully review the assignment and make a backlog of everything they can possibly think of that they will need to do in order to create a successful final product. You might do this twice—once before the students meet with their service-learning community partner and once after.
- *Introduce the new teams to Agile project language.* Once students have created a list, you might talk a bit about Agile, explaining that they will be trying some new project management strategies and briefly describing how Agile is used by cross-functional teams in organizations at which they may soon find work. Then introduce the concepts of stories and tasks, asking them to reassess and

reorganize their to-do list by creating stories and adding appropriate tasks under the stories, taking into account any process documents or interim deadlines required in the assignment. You can walk around the classroom providing advice and suggestions and pointing out places where stories can be broken down further or combined. It is less important for them to nail every story than it is for them to begin to articulate the parts of the process that they must accomplish in order to complete the project with the community partner.

- *Introduce a simple Scrum folder approach.* Provide teams with small sticky notes, plain manila folders, pens, and rulers,³ helping them to set up the three-column Scrum boards. Have students then write their stories and tasks on individual sticky notes and move them to the first column, the backlog. Explain how they can prioritize the stories and tasks by moving them up and down the column and show them how to choose priority tasks that they will place in the WIP column. You might also discuss the WIP column, helping them to understand the value of committing only to work activities that they can finish between team meetings. Finally, explain how to use the folders throughout the project in order to visualize and make consistent progress on their project goals.

You might also discuss how you will use the folder in terms of monitoring progress or grading. I read updated folders after every class, making notes about each team's progress and any questions I wanted to ask. You might use it more formally to keep track of a participation grade. But after providing those final explanations, turn the project over to the teams and let them get started. If the teams use the folders to prioritize tasks and manage their projects, they will most likely experience both successes and setbacks, both of which provide just-in-time teaching moments.

All of the strategies can also be used in longer term or full-semester projects such as the grant writing course that I described. But to more fully jump into the Agile pool, I recommend these additional strategies:

- *Frame the project in defined sprints.* Depending on your preference or the students' skill level, you can articulate the course into four or five sprints with interim products that you designate or allow the students to break the project into their own sprints after an initial planning meeting. The sprint cycle can be used to hold students accountable for regularly creating valuable pieces of a

project and help them to learn to more effectively estimate tasks and manage project time. You can support the groups by facilitating, coaching, mentoring, and holding students to the process as necessary.

- *Hold regular grooming and planning meetings.* Scrum is plan emergent rather than plan driven, which gives teams the responsibility of not only understanding the project's big picture but also responding effectively to change during the project. Hold grooming meetings at the start of each sprint, before sprint planning, to allow students to revisit the stories and tasks that they originally planned, adding and deleting stories as necessary based on the last sprint and reprioritizing the backlog accordingly. Planning meetings can be held immediately after grooming so that students can let their plan emerge by deciding on stories and tasks that they will commit to for the next sprint based on their new priorities.
- *Require daily Scrum team meetings.* Agile coaching guru Adkins (2010) described daily Scrum meetings as commitment rather than progress meetings. By regularly reporting back to their team exactly what they each have accomplished, are currently working on, and might need help with, team members learn to be and hold each other accountable while they keep up their team's momentum and energy. Teams can use this time to update their Scrum board or folder, set intermediate goals, and address pressing challenges. As I have found, once students get comfortable with daily Scrums, they seem to become more comfortable addressing with each other issues such as social loafing, miscommunication, and quality control, thereby improving their communication and collaboration skills.
- *Invest time in reviews and retrospectives.* Just as grooming and planning meetings set the tone for a sprint, reviews and retrospectives provide the opportunity for students to demonstrate their progress and carefully reflect on their process. In a review meeting, ask students to showcase their sprint work for their team or stakeholders, including the client or community partner, in order to get feedback. In a retrospective, ask the teams to reflect on what did and did not go well during the last sprint, allowing teams the opportunity to provide honest feedback and commit to improving some aspect of the process for the next sprint. A great deal of learning can happen in these meetings as students mature in the process as writers and collaborators.

As in industry, Agile and Scrum can be difficult to fully implement in the classroom, and based on my experience over the last 5 years, Scrum is not useful in all situations. For example, I would caution against using the Scrum framework on very short projects, especially ones that only require students to coordinate individual work or cooperate for a short period of time. Scrum was designed for complex, multifaceted projects that require close collaboration; if team members do not need to collaborate, being tied to the framework can seem like busywork. Aspects of Agile practices such as task articulation and visualization can still be useful for these groups, but beyond that, I recommend teaching Agile within the context of more complex collaborative projects that allow teams to dig into the strategies more authentically and for which teams must rely on the input of every team member for project success.

While Agile and Scrum have become increasingly popular in the last 15 years, traditional project management schemes still hold the market share in most industries. Yet learning aspects of Agile and Scrum processes can help students develop a stronger toolkit of project management strategies that they can draw on in the future regardless of what workplace environment they enter. Technical communicators in the field are already reporting that in successful Agile environments, they are better able to advocate for the user and develop strong external documentation throughout the development process. By allowing students to practice both traditional, linear development and Agile strategies, we can better prepare prospective technical and professional writers for their futures as user advocates and document leaders.

Declaration of Conflicting Interests

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This project was supported by funding from the Elon University Center for the Advancement of Teaching and Learning (CATL) Fellowship program.

Notes

1. Whether or not Agile should be capitalized is an ongoing debate within the industry, and both forms can be found in professional publications. I have chosen to

capitalize Agile throughout to denote a specific set of practices rather than a general way of acting.

2. This study was approved by my university's institutional review board, and I followed closely the Scholarship of Teaching and Learning protocols for collecting student data.
3. I prefer the folder method for the psychological boost people get from physically moving tasks into the Done column. But in different environments, Word files in a Dropbox, Google spreadsheets, or project management software are all potentially viable options.

References

- Adkins, L. (2010). *Coaching Agile teams: A companion for ScrumMasters, Agile coaches, and project managers in transition*. Boston, MA: Addison Wesley.
- Alfonso, M., & Botia, A. (2005). *An iterative and Agile process model for teaching software engineering*. Paper presented at the 18th CSEET, Ottawa, Ontario, Canada.
- Allen, N., & Benninghoff, S. T. (2004). TPC program snapshots: Developing curricula and addressing challenges. *Technical Communication Quarterly*, 13, 157–185.
- Anderson, D., Augustine, S., Avery, C., Cockburn, A., Cohn, M., DeCarlo, D., Fitzgerald, D., . . . Wysocki, R. (2005). *The declaration of interdependence for modern management or DOI*. Retrieved from <http://alistair.cockburn.us/The+declaration+of+interdependence+for+modern+management+or+DOI>
- Austin, G. (2012). *How writers can thrive in Agile software development*. Retrieved from <http://www.slideshare.net/gavaustin/how-writers-can-thrive-in-agile-software-development>
- Austin, G., & Berry, M. (2011). *A writer's guide to surviving Agile software development*. Retrieved from <http://www.scrumalliance.org/community/articles/2011/september/a-writer-x27-s-guide-to-surviving-agile-software-d>
- Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). *Manifesto for Agile software development*. Retrieved from <http://www.agilemanifesto.org>
- Beedle, M. (2006). *Essential Agile characteristics*. Retrieved from <http://c2.com/cgi/wiki?EssentialAgileCharacteristics>
- Benson, J., & Barry, T. D. (2011). *Personal Kanban: Mapping work | Navigating life*. Seattle, WA: Modus Cooperandi Press.
- Brown, J. K., & Chao, J. T. (2012). Collaboration of two service-learning courses: Software development and technical communication. *Issues in Informing Science and Information Technology*, 7, 403–412.

- Cleland, S. (2003). *Agility in the classroom: Using Agile development methods to foster team work and adaptability amongst undergraduate programmers*. Paper presented at the 16th Annual NACCQ Conference, Palmerston North, New Zealand.
- Cohn, M. (2010). *Succeeding with Agile: Software development using Scrum*. Boston, MA: Addison-Wesley.
- Cohn, M. (2011, October 24). *User stories, epics, and themes [Web log message]*. Retrieved from <http://www.mountaingoatsoftware.com/blog/stories-epics-and-themes>
- Coker, L. S. (2010). [Review of the book *Agile coaching*, by R. Davies]. *Technical Communication*, 57, 432–432.
- Dayton, D., & Barnum, C. (2009). The impact of Agile on user-centered design. *Technical Communication*, 56, 219–234.
- Dennings, S. (2012, April 17). *The case against Agile: Ten perennial management objections*. Retrieved from <http://www.forbes.com/sites/stevedenning/2012/04/17/the-case-against-agile-ten-perennial-management-objections/>
- Dicks, R. S. (2003). *Management principles and practices for technical communicators*. New York, NY: Longman.
- Dicks, R. S. (2013). How can technical communicators manage projects? In J. Johnson-Eilola & S. A. Selber (Eds.), *Solving problems in technical communication* (pp. 310–332). Chicago, IL: University of Chicago Press.
- Fox, A., & Kramer, M. (2008) *Mobile and Agile: The floating writer's survival kit*. Retrieved from <http://www.writersua.com/articles/AGILE/index.html>
- Freedman, R. (2009, June 16). *The roots of Agile project management [Web log message]*. Retrieved from <http://www.techrepublic.com/blog/tech-manager/the-roots-of-agile-project-management/1491>
- Grant, T. (2013). *Agile in the real world: Gone mainstream, creating bigger waves, making course corrections*. Retrieved from <http://www.slideshare.net/Tom-GrantForr/agile-2013-presentation-tom-grant>
- Hackos, J. T. (2006). *Information development: Managing your documentation projects, portfolio, and people*. Indianapolis, IN: John Wiley.
- Highsmith, J. (2001). *History: The Agile manifesto*. Retrieved from <http://www.agilemanifesto.org/history.html>
- Hislop, G. W., Lutz, M. J., Naveda, F., McCracken, W. M., Mead, N. R., & Williams, L. A. (2002). Integrating Agile practices into software engineering courses. *Computing Science Education*, 12, 169–185.
- Lingard, R., & Barkataki, S. (2011). *Teaching teamwork in engineering and computer science*. Paper presented at the 41st ASEE/IEEE Frontiers in Education Conference, Rapid City, SD.
- McNely, B., Gestwicki, P., Burke, A., & Gelms, B. (2012). Articulating everyday actions: An activity-theoretical approach to Scrum. In *SIGDOC '12: Proceedings*

- of the 30th Annual Conference on Design of Communication (pp. 95–104). New York, NY: ACM. Retrieved from <http://dl.acm.org/citation.cfm?id=2379057&pickeid=prox&cfid=544110266&cftoken=96745940>
- Melnik, G., & Maurer, F. (2003). Introducing Agile methods in learning environments: Lessons learned. In *Extreme Programming and Agile Methods-XP/Agile Universe 2003* (pp. 172–184). Berlin, Germany: Springer.
- Meloncon, L., & Henschel, S. (2013, February). Current state of U.S. undergraduate degree programs in technical and professional communication. *Technical Communication*, 60, 45–64.
- Miller, C. (1979). A humanistic rationale for technical writing. *College English*, 40, 610–617.
- Miller, C. (1989). What's practical about technical writing? In B. E. Fearing & W. K. Sparrow (Eds.), *Technical writing: Theory and practice* (pp. 14–24). New York, NY: Modern Language Association.
- Miller, T. (1991). Treating professional writing as social praxis. *Journal of Advanced Composition*, 11, 57–72.
- Moore, J. (2013, May). *Documentation thrives in an Agile methodology*. Paper presented at the Society for Technical Communication Summit 2013, Atlanta, GA.
- Perera, G. I. U. S. (2009). Impact of using Agile practice for student software projects in computer science education. *International Journal of Education and Development Using Information and Communication Technology*, 5, 85–100.
- Pope-Ruark, R. (2012). “We Scrum every day”: Using scrum project management framework for group project. *College Teaching*, 60, 164–169.
- Pope-Ruark, R. (2014). A case for metic intelligence in technical and professional writing programs. *Technical Communication Quarterly*, 23, 323–340. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/10572252.2014.942469#.U_tYpmPCd-R doi:10.1080/10572252.2014.942469
- Pope-Ruark, R., Eichel, M, Talbott, S., & Thornton, K. (2011, Spring). Let's Scrum: How Scrum methodology encourages students to view themselves as collaborators. *Teaching and Learning Together in Higher Education*. Retrieved from <http://teachingandlearningtogether.blogs.brynmawr.edu/archived-issues/may-issue/lets-scrum>
- Reichlmayr, T. (2003). An Agile approach to an undergraduate software engineering course project. *Frontiers in Education*, 3, 13–18.
- Rubin, K. S. (2012). *Essential Scrum: A practical guide to the most popular Agile process*. Boston, MA: Addison-Wesley.
- Schwaber, K., & Sutherland, J. (2011). *Scrum guide*. Retrieved from http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf
- ScrumAlliance. (n.d.). *The benefits of Scrum*. Retrieved from http://www.scrumalliance.org/pages/benefits_of_scrum

- Sigman, C. M. (2010). *Agile development*. Retrieved from <http://stcbok.editme.com/Agile-Development>
- Slaten, K. M., Droujkova, M., Berenson, S. B., Williams, L., & Layman, L. (2005). Undergraduate student perceptions of pair programming and Agile software methodologies: Verifying a model of social interaction. In *Proceedings of the Agile Development Conference (ADC '05)*, pp. 323–330. Washington, DC: IEEE.
- Szalvay, V. (2004.) *An introduction to Agile software development*. Retrieved from http://www.danube.com/docs/Intro_to_Agile.pdf
- Vaishampayan, V. (2012). [Review of the book *The Agile Samurai: How Agile masters deliver great software*, by J. Rasmusson]. *Technical Communication*, 59, 151.

Author Biography

Rebecca Pope-Ruark is an associate professor of English, specializing in professional writing and rhetoric, at Elon University. She teaches courses in rhetorical theory, publishing, writing research, and project management. Her research centers on Agile practices in the classroom, faculty professional development, and scholarship of teaching and learning.

Copyright of Journal of Business & Technical Communication is the property of Sage Publications Inc. and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.