SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education, Association for Computing Machinery, 2020, Pages 643–649 https://doi.org/10.1145/3328778.3366864

What Agile Processes Should We Use in Software Engineering Course Projects?

An Ju
an_ju@berkeley.edu
EECS, University of California, Berkeley
Berkeley, California, USA

Adnan Hemani
adnan.h@berkeley.edu
EECS, University of California, Berkeley
Berkeley, California, USA

Yannis Dimitriadis
yannis@tel.uva.es
Universidad de Valladolid
Valladolid, Spain

Armando Fox
fox@cs.berkeley.edu
EECS, University of California, Berkeley
Berkeley, California, USA

ABSTRACT

While project-based software engineering courses aim to provide learning opportunities grounded in professional processes, it is not always possible to replicate every process in classrooms due to course constraints. Previous studies observed how students react to various processes and gave retroactive recommendations. In this study, we instead combine a field study on professional Agile (eXtreme Programming, XP) teams and an established team process taxonomy to *proactively* select team processes to incorporate in a project-based software engineering course. With collected knowledge from the field study, we choose three XP processes to augment the design of a mature software engineering project course. We choose processes that are 1) considered important by professionals, and 2) *complete* with respect to coverage of the taxonomy's main categories. We then compare the augmented course design with the original design in a case study. Our results suggest that 1) even without extra resources, adding these new processes does not interfere with learning opportunities for XP processes previously existing in the course design; 2) student teams experience similar benefits from these new processes as professional teams do, and students appreciate the usefulness and value of the processes. In other words, our approach allows instructors to make conscious choices of XP processes that improve student learning outcomes while exposing students to a more *complete* set of processes and thus preparing them better for professional careers. Course designers with limited resources are encouraged to use our methodology to evaluate and improve the designs of their own project-based courses.

KEYWORDS

software engineering; Agile; project-based learning

ACM Reference Format:
An Ju, Adnan Hemani, Yannis Dimitriadis, and Armando Fox. 2020. What Agile Processes Should We Use in Software Engineering Course Projects?. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366864

1 INTRODUCTION

Project-based software engineering courses aim to provide learning opportunities grounded in practice [23, 29]. They emphasize processes that are used by professionals to work smoothly and efficiently in teams [20, 24, 29, 34]. In this study, we focus on the widely-used Agile software development methodology [10], and specifically, the eXtreme Programming (XP) method [2].

Ideally, instructors want to teach students *all* XP processes that they may use as professionals [11, 29]. However, with limited resources and various course constraints, instructors must decide *which* processes to include and *how* to adapt them for classrooms [4, 11, 21]. The decision should not be arbitrary; in this study, we choose as our main constraint that ***the selection of processes should be* complete *with respect to a well-established taxonomy.*** And we present a method for instructors to make choices and adaptations informed by professional processes.

In contrast to "retroactive" methods, which may allow undesired outcomes, we rely on an established team process taxonomy of Marks et al. [19], which shows three categories of team processes essential for any working teams, including software development teams. We combine the taxonomy with professional processes at a software company through a field study. By mapping both professional processes and our current pedagogy into the taxonomy, we identify areas in our course design that are under-covered. Then, we augment the course design with XP processes that are important for professionals and *complete* for working teams with respect to the taxonomy's main categories. With this method, we improve our course's completeness without risking undesired outcomes.

Our case study examines two research questions about the augmented course design. First, as both students and instructors have limited resources, we investigate whether new processes interfere with learning opportunities for XP processes that are previously included in the course design. Second, we ask whether the new processes deliver benefits to the students comparable to those experienced by professional teams, and whether students recognize the usefulness and value of the new processes. As studies have shown, students may fail to recognize the value of a process if it is improperly used in classrooms [3, 11]. Our case study shows positive results for both questions by comparing self-reported performance from this case study with the most recent previous offering of the same course. These results suggest that our method could achieve desired learning objectives while avoiding undesired outcomes such as overloaded students.

The contributions of this paper are therefore as follows:

(1) A method for using a team process taxonomy as the basis for measuring the *completeness* of process coverage in a software project course and making informed choice of processes without experiments;
(2) A field study in an XP software company, allowing us to choose which processes to incorporate into an existing XP project course;
(3) Initial results of integrating three selected processes into a course, suggesting that processes selected with our approach improve learning outcomes.

The rest of the paper is organized as follows. Section 2 compares our approach with previous studies and briefly reviews the team process theory literature. Section 3 describes our field study of a successful, mature professional organization, explaining how XP processes map to the taxonomy and allowing us to choose three processes to fix gaps in our pedagogy. Section 4 describes the results of incorporating the three processes into the course, validating the benefits of the augmented course design. Section 5 discusses threats to validity and future work.

2 RELATED WORK

Several studies and experience reports have shown that the academic environment has constraints that prevent replicating industrial Agile processes in classrooms [3, 5, 9, 30, 36]. For example, it may be impossible for students to meet at a time other than scheduled lectures [3]. Thus, instructors should carefully determine what processes students should follow and make necessary adaptations [11].

Masood et al. [21] examined course constraints, adaptations that students made to Agile processes, and effects of those adaptations. They recommended all Agile processes to students and observed how students adapted processes due to course constraints. It was unclear, however, whether and how adaptations made by students influenced learning outcomes. In contrast, we proactively adapt processes based on industrial observations and a team process taxonomy. We believe this method leads to a course design that has a firm real-world grounding [23] and clear learning benefits designed by *instructors*.

Similarly, Goldman et al. presented an experience report about teaching XP in classrooms [11]. The authors found that adaptations may be needed because not all XP practices are possible or desirable in all situations. They made retroactive recommendations based on their experiences. In contrast, instructors can use our method to make informed choices and adaptations without experimenting on real students, thus reducing the risk of undesired outcomes.

We note that that evolving teaching methods *with* Agile or XP [28, 35] is out of this paper's scope. Instead, we examine methods *for* teaching Agile/XP. The focus of our study is an approach that allows instructors to make informed design choices that benefit students and emphasize industrial practices. We focus on project-based learning as this is a widely used and effective teaching method for software development.

We inform our selection of Agile processes using team process theories. Moe et al. [25] used a team process theory to observe a team transitioning to the Scrum methodology [29]. They were able to connect observed barriers to success with the theory, showing that team process theories could explain the failure and success of Agile teams. Similarly, Lindsjørn et al. [18] borrowed ideas from teamwork theories to examine the success factors in Agile software development. They designed a survey based on a teamwork theory to measure teamwork quality and team success. Based on the survey data, they showed that teamwork is a significant predictor of team performance. The two studies showed that team theories could provide useful insights and measurements for Agile teams. In this study, we choose Marks et al.'s team process taxonomy [19]. We recognize that there are other successful team process theories [6, 15, 33], but they do not contradict Marks et al.'s taxonomy. Instead, they present alternative perspectives that confirm and extend the taxonomy. Thus, this taxonomy can serve our goal of selecting processes that are *complete* for working teams, since the format of taxonomy gives us a straightforward way to define measurements for completeness.

3 OBSERVE, CLASSIFY, AND CHOOSE XP PROCESSES

Pivotal[1] is a US-based company that builds software for clients using the XP, works with clients to transition their organizations to XP, and provides services and tools to support XP. The company has collaborated with other software engineering researchers [32].

We observed 3 Pivotal teams of different maturity levels. Our goal was to observe processes that teams used in an effective workflow and classify those processes with a taxonomy. The eventual goal of doing so was to identify a subset of processes for incorporating into our XP course according to the following criteria:

- The process is judged important by professionals.
- The process fills a gap in our course design.

3.1 Data collection

We use observations, interviews, and a survey to collect data. Data is collected between May 2018 and September 2018.

Observations. We conducted observations on 3 teams for about one week each. The company uses 1-week XP iterations, so observing a team for one week allows us to observe all the processes they use during a typical iteration. We also participated in a few meetings of other teams. In the observations, we started with a broad interest about workflows and focused on a subset of behaviors/meetings that were most relevant to the taxonomy (listed in Figure 1) as the research proceeded.

Interviews. For data triangulation, we conducted 5 semi-structured interviews with 4 employees (1 product manager, 3 engineers).

[1]https://pivotal.io

Category	Process	Definition	Importance
Transition Process	Pre-IPM	A meeting prior to an IPM where stories are prepared.	1.8
	Iteration Planning Meeting (IPM)	Create an iteration plan [2, 27, 31].	2.8
Action Process	Backlog	Creat, prioritize, implement, and accept/reject user stories [2, 31].	3.0
	Continuous Integration (CI)	Frequent integration of individual's code into a mainline [2].	3.4
	Information Radiator	Set up a monitor to track key information [2].	2.1
	Pairing	Pair programming applied to coding, managing, and design works [2].	2.9
	Standup	Short daily meetings [31].	2.4
	Test-Driven Development (TDD)	Write tests before the production code [2].	3.2
Interpersonal Process	**Retrospective**	Discuss team problems openly at the end of an iteration [27, 31].	3.5

Figure 1: Categories in the taxonomy, XP processes, explanations, and their importance. We choose IPM, Retrospective, and Standup (all highlighted in the table) to examine in the case study. Importance of a practice is averaged over survey responses (0="Not at all important", 1="Slightly important", 2="Somewhat important", 3="Very important", 4="Extremely important").

Survey. We designed a 12-question survey asking opinions about the importance of different processes within Pivotal [2]. A contact within Pivotal distributed the survey internally to all developers via an email list. We received $N = 18$ responses.

3.2 Analysis: Transition, Action, Interpersonal Processes

Marks et al.'s taxonomy of team processes [19] is developed from studies on general teams, and distinguishes *transition, action,* and *interpersonal* processes, among others:

- Transition processes are ones in which "teams focus primarily on evaluation and/or planning activities to guide their accomplishment of a team goal or objective."
- Action processes are "periods of time when teams conduct activities leading directly to goal accomplishment."
- Interpersonal processes "typically lay the foundation for the effectiveness of other processes."

We use a closed coding method to explain our observations, with codes derived from the taxonomy. Observation notes, interview notes, and transcripts are coded line-by-line. One researcher coded the data alone. Figure 1 lists all the processes we observed and which of the three categories they fall into. Processes with names **in bold** are the ones we decided to incorporate into our course, and which we describe in more detail in the rest of this section.

IPM is a transition process. An Iteration Planning Meeting (IPM) occurs at the beginning of each iteration. One objective of the IPM is to develop a shared understanding of the value, constraints, and acceptance criteria of each *user story*, the basic unit of work during an iteration, "to ensure everyone is on the same page and has a common understanding of the stories in the backlog" (as one survey response states). In the IPMs we observed, one person usually explained each user story, the team clarified each story's expectations through explanations and discussions, and the meeting would not proceed until all developers indicated that the expectations were clear. Some details of this discussion might be recorded in the story's description. IPM is directly linked to planning; therefore, it is a transition process.

[2]https://github.com/ace-lab/xp-study-sigcse20/blob/master/PivotalProcessSurvey.pdf.

Standup is an action process. A primary objective of Standup is to discuss new findings and blocking factors. Standup is a short (a few minutes) daily meeting in which members take turns to update their status and plan. One survey response found Standup to be "most effective as a daily 'health check' and context-sharing activity." One interviewed engineer mentioned that "problems would be surfaced by a Standup and thus other team members can help to fix them." Standup directly contributes to the team's goal accomplishment by unblocking the team and sharing team status. Therefore, Standup is an action process.

Retrospective is an interpersonal process. The Retrospective occurs at the end of each iteration, after the iteration's work has been delivered. During a Retrospective meeting, the team's practices are analyzed and improved. As one interviewed engineer put it, "[Retrospective] is the only practice that allows the team to tweak the practices" and "Retrospective is the place where you analyze your process and try to make it better." Furthermore, Retrospective regulates team emotions. In a typical retrospective, members express feelings on a message board, physical or virtual, with three columns for different feelings (a standard set of choices is "happy, meh [idiomatic American English for indifferent], and sad"). Retrospective is thus a place where team members can safely share their feelings for the team to stay effective in other processes. One inteviewed engineer mentioned that "creating that space where everybody on the team feels like they can say things that they are feeling" is important for Retrospective. In summary, Retrospective lays the foundation for teams to stay productive, and thus, Retrospective is an interpersonal process.

3.3 Select Processes

In selecting processes, we first exclude those that are already covered in previous and current offerings of the course: *Backlog*, *CI*, *TDD*. (We describe the course more completely in Section 4.) We exclude *Information Radiator* and *Pre-IPM* because their importance as judged by professional practitioners is low (as shown in Figure 1), and because we do not want to overwhelm students with too many processes in a single course. While professional practitioners we interviewed practice *Pairing* exclusively, previous studies suggest that pairing may be impractical to do consistently in classrooms [3, 26]. As such, pairing was recommended but not mandated, as in previous offerings of the course.

This leaves *IPM*, *Retrospective*, and *Standup* to focus on in our case study. We note that in addition to being missing from previous offerings of the course, these three processes provide students with learning opportunities in all three categories of Marks et al.'s taxonomy: *IPM* is a transitional process, *Standup* is an action process, and *Retrospective* is an interpersonal process. Therefore, adding these processes improves the *completeness* of the course from the perspective of covering the taxonomy.

4 CASE STUDY

This case study was conducted at the University of California, Berkeley, from January through May 2019, in the context of CS169, a 14-week project-centric software engineering course that teaches XP. The course had 120 students divided into 20 teams. Each team worked with a customer to develop or enhance a Web application tailored to the customer's small-business need.

During an initial short iteration, student teams met with their customer to understand the requirements and goals for the project. The teams then worked on the project during four 2-week iterations.

The course was supported by two 20-hour-per-week teaching assistants, each of whom supervised and coached 10 student teams, meeting with each team during each iteration to provide feedback and evaluation. Both teaching assistants were also researchers in this study.

We compare this case study with the most recent offering of the same course. The previous course was offered in Fall 2017. It had 126 students divided into 21 teams. The course had the same course project schedule. In this case study, we made the following major updates to the course design:

- Added checkpoints for IPM, Customer Meeting, and Retrospective in order to incorporate the processes.
- Updated the self-assessment survey to track newly incorporated processes.
- Gave students access to a new metrics dashboard that was different from the one used previously. We made this change migrating the dashboard to a new platform [12]. The new dashboard presented a similar set of metrics about Backlog, CI, and TDD as the previous one. Thus, we expected this change to have a minimal impact on this case study.
- Gave instructors access to data from newly added tools. Although the data was not used for grading, instructors could use it to provide feedback in meetings.

The case study focuses on investigating whether students benefit from these augmentations. Specifically, we focus on the following two questions:

- Do new processes interfere with learning opportunities provided by processes already present in the course?

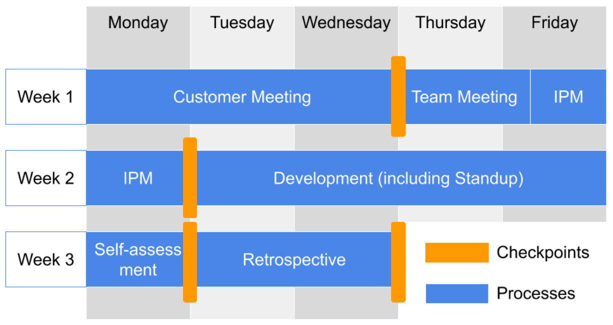


Figure 2: Iteration schedule. The development stage includes other processes such as CI, Backlog, and Standup.

- Do students appreciate the usefulness and values of new processes for XP development?

4.1 Embed Processes

Figure 2 is the iteration schedule. Each iteration is two weeks long, but we leave some overlap so that students have enough time for retrospectives. Thus, the start of week 3 is also the start of the next iteration.

Because teaching processes is demanding for instructors [8, 13, 24, 29], new processes are supported by student-facing tools. The tools give instructors more visibility into new processes with usage data and allow students to learn new processes with more support. We adapt open-source tools *planning poker* [3], *postfacto* [4], and *slack manager* [5] for IPM, Retrospective, and Standup respectively.

Besides tools, we use checkpoints to make processes transparent to students. Checkpoints are online surveys that students complete every iteration [6]. Checkpoints make instructor expectations explicit so that students can follow and improve over time. In our case study, checkpoints are mandatory and graded.

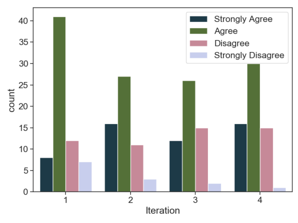
For each iteration, we changed two or three questions for teaching purposes. For example, in Iteration 4 we added two questions each to the IPM and Retrospective checkpoints. One question asks how confident the student is in contributing to a productive process (IPM or Retrospective) in a professional team. The other asks how likely the student would be to use this process (IPM or Retrospective) in the future if the student were leading a team.

4.2 Data Collection

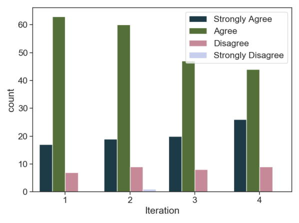
We collect data about tool use from checkpoints, and data about learning outcomes from checkpoints and self-assessment surveys [7].

The self-assessment survey contains both questions used in previous courses and questions newly added in this case study. For example, we add questions about team communication to measure the effect of Standup. The survey is adapted from surveys used

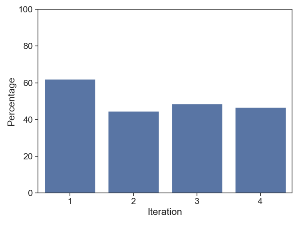
[3]https://github.com/bimovidia/planning-poker
[4]https://github.com/pivotal/postfacto
[5]https://github.com/anonrig/slack-manager
[6]IPM: https://github.com/ace-lab/xp-study-sigcse20/tree/master/IPM. Retrospective: https://github.com/ace-lab/xp-study-sigcse20/tree/master/Retrospective.
[7]https://github.com/ace-lab/xp-study-sigcse20/tree/master/Self-assessment.



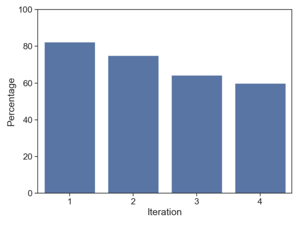
(a) Usefulness of *planning poker*



(b) Usefulness of *postfacto*



(c) Use of *planning poker*



(d) Use of *postfacto*

Figure 3: Tool use and tool satisfaction reported by students ($107 \leq N \leq 117$ for all iterations).

and validated in previous studies, focusing on XP processes [16], software development waste [32], and teamwork [14].

We acknowledge that our data is not an objective measurement of learning outcomes. However, studies show that self-efficacy—students' belief and self-confidence in their capabilities and in their ability to use a particular course of action to achieve a desired result [1]—is a reasonable predictor of student performance [7]. Our self-assessment surveys and checkpoints measure self-efficacy, so we use them as reasonable proxy measurements for learning outcomes. Previous studies have also shown that surveys are useful instruments for teaching software engineering processes [22].

4.3 Results

Students followed the processes and were satisfied with the tools supporting them. Figure 3 presents student-reported tool usefulness and tool usage. Although most students agree that the tools (*planning poker* and *postfacto*) are useful for their team development, Figures 3c and 3d show that only 40%-60% of students used *planning poker* and 50%-80% used *postfacto*). Further investigation showed that in most cases, students used an alternative approach for the process. For example, students reported in Iteration 4 that they used in-person discussions instead of the tool. One student mentioned that "we implemented this into our general meeting pretty naturally, so we didn't really need it."

Iteration 4's survey results suggested that students had proper training on IPM and Retrospective with the new course design. 41 (38%) students reported that they were "extremely confident" in contributing to a productive IPM in a professional team while 48 (45%) reported that they were "very confident"; 41 (37%) students reported that they were "extremely confident" in contributing to a productive retrospective meeting in a professional team while 51 (46%) reported that they were "very confident."

The inclusion of new processes does not interfere with existing learning opportunities. In this case study, students achieved similar or better self-reported performance on all previously tracked XP processes. Presented in Figure 4a to Figure 4e, the median of self-reported performance on these tracked processes in this case study (Sp19) is the same or higher than the previous semester (Fa17) in Iteration 4. Furthermore, with an alternative hypothesis that Iteration 4's score in this case study is higher than the score of the previous semester, a Mann-Whitney rank test shows that student performance is significantly better on Planning, Pairing, and Technical Debt ($p \leq 0.01$). This result indicates that the augmented course design does not negatively influence existing processes.

Students appreciate the usefulness and value of the new processes (IPM, Retrospective, Standup) for XP development. Planning improves significantly in this case study (Figure 4a). This is directly linked to IPM. In the Iteration 4 survey, 58 (54%) respondents would "definitely" use IPM if leading a team and 41 (38%) "probably" would, suggesting that most students appreciate the usefulness of IPM for team development.

Improvements (value differences) measured by self-assessment surveys between two consecutive iterations are significantly higher in this case study than the previous semester for all processes except for *Pair programming* and *Planning* (Mann-Whitney rank test, $p \leq 0.05$). This observation indirectly supports the claim that Retrospective has changed student behaviors, since Retrospective focuses on process improvements. When asked in Iteration 4 whether they would use Retros if leading a team in the future, 51 (46%) responded "definitely" and 49 (44%) responded "probably", which suggests that most students appreciate the usefulness of Retrospective for team development.

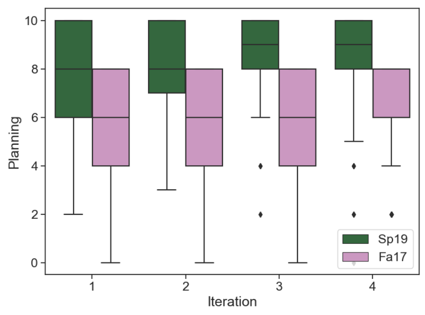
Figure 4f shows that teams have a high communication score across iterations, which is related to the use of Standup. For comparison, according to one study[14], professional software teams answering the same questions report an average of 0.84 (normalized) for communication.

In summary, self-assessment surveys show improvements that are related to *IPM*, *Retrospective*, and *Standup*. It suggests that student teams benefit from the processes as professional teams. Iteration 4's survey responses support the inference that students appreciate the value of the processes.

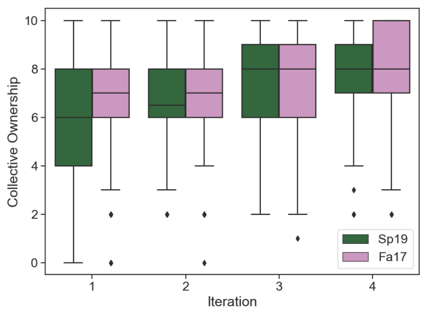
Summary. When we changed our project-based course to incorporate *IPM*, *Standup*, and *Retrospective* selected in Section 3, we found that in comparison with a previous offering of the course lacking these processes: (1) students reported similar or better performance on existing XP processes, suggesting that our design augmentation does not interfere with existing processes; (2) student teams can benefit from new processes as professional teams, and students recognize the usefulness and value of these processes. Thus, we believe our approach can result in augmented course designs that lead to better learning outcomes.

5 DISCUSSION

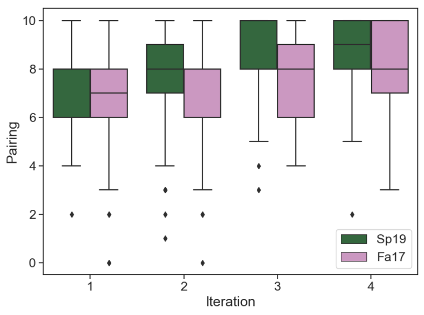
Threats to validity. Field observations conducted in one company, however mature, do not necessarily represent the industry; researchers can apply our method to observe other companies so as



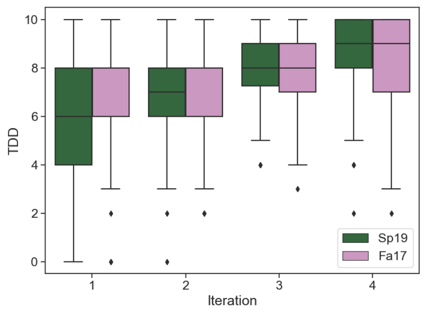
(a) Planning



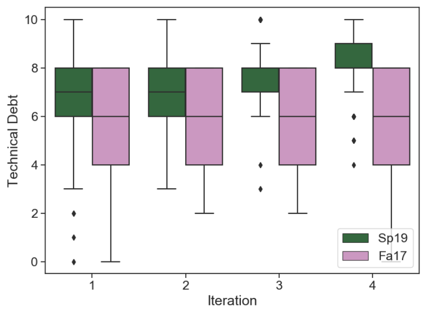
(b) Collective Ownership



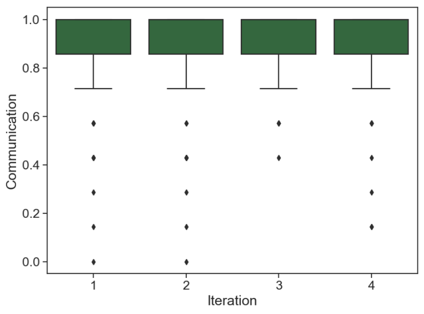
(c) Pairing



(d) Test-driven Development



(e) Technical Debt



(f) Communication

Figure 4: Self-reported process performance (communication is not previously measured). $112 \le N \le 115$ **for all iterations.**

to analyze more industrial practices. That said, Pivotal is among the most disciplined and consistent practitioners of XP, having created widely-used tools including the Jasmine JavaScript TDD framework and the Tracker project manager to support their workflows and processes. Also, while we used a field study to emphasize the industrial grounding of our selection of processes, practitioners may rely on other data sources for this purpose.

We rely on one *in vivo* case study to examine the new processes. The "gold standard" of controlled A/B testing is challenging to achieve in such a setting. First, we do not have the resources to manage enough student teams for a statistically reliable comparison. Second, student teams work on different projects, which is a confounding factor that we cannot avoid. Third, withholding potentially pedagogically-helpful tools or processes from a control group raises ethical issues in a setting where students must receive course grades. Thus, we chose case study as our research method, despite these confounding factors that might influence student behaviors and self-assessments.

This paper focuses on XP. Other Agile variants such as Scrum [29] also exist. We do not claim XP is superior, but we choose it because the course examined in Section 4 focuses on XP. Our method is generalizable to other Agile variants, since all Agile methods stem from a similar set of values and often share processes. Indeed, *IPM*, *Retrospective*, and *Standup* are also important processes in Scrum.

Future work. Our case study results do not show a causal relationship between the augmented course design and improved learning outcomes. More case studies using our method and carefully designed controlled experiments could provide more understandings to XP and Agile processes in classrooms. In addition, our measurements of learning outcomes rely on self-reported surveys. While these are reasonable proxy measurements, future studies should evaluate learning outcomes with objective measurements, such as code quality and velocity [17].

6 CONCLUSION

Combining an established taxonomy with a field study, we selected three processes to augment a course design, and showed evidence that these additions improve learning outcomes without interfering with prior processes. Teaching and measuring team processes is difficult [8, 22], but using tools that generate instructor-visible data, as we have done, can provide better insights into student behaviors. In the future we hope to use this data to determine whether teams that adhere more closely to the processes experience better learning outcomes, produce higher quality software artifacts, or both.

We believe we are among the first to describe and follow this systematic approach to incorporating team processes into XP pedagogy. Compared with previously reported retroactive approaches, our approach proactively informs the course design, avoiding designs that are overwhelming for students [29] or in which students fail to appreciate the value of the processes being taught [26].

ACKNOWLEDGMENTS

The research of the third author is partially funded by the European Regional Development Fund and the National Research Agency of the Spanish Ministry of Science, Innovations and Universities under project grants TIN2017-85179-C3-2-R, by the Ministry of Science and Education (PRX17/00410), by the European Regional Development Fund and the Regional Council of Education of Castile and Leon under project grant VA257P18, and by the European Commission under project grant 588438-EPP-1-2017-1-EL-EPPKA2-KA

REFERENCES

[1] Albert Bandura, Claudio Barbaranelli, Gian Vittorio Caprara, and Concetta Pastorelli. 2001. Self-efficacy beliefs as shapers of children's aspirations and career trajectories. *Child development* 72, 1 (2001), 187–206.

[2] Kent Beck and Erich Gamma. 2000. *Extreme programming explained: Embrace change.* Addison-Wesley Professional.

[3] Joe Bergin, James Caristi, Yael Dubinsky, Orit Hazzan, and Laurie Williams. 2004. Teaching software development methods: the case of extreme programming. In *ACM SIGCSE Bulletin*, Vol. 36. ACM, 448–449.

[4] Jennifer Campbell, Stan Kurkovsky, Chun Wai Liew, and Anya Tafliovich. 2016. Scrum and Agile Methods in Software Engineering Courses. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*. ACM Press. https://doi.org/10.1145/2839509.2844664

[5] David Delgado, Alejandro Velasco, Jairo Aponte, and Andrian Marcus. 2017. Evolving a Project-Based Software Engineering Course: A Case Study. In *2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. https://doi.org/10.1109/cseet.2017.22

[6] Terry L Dickinson and Robert M McIntyre. 2010. A Conceptual Framework for Interprofessional Teamwork. In *Interprofessional Teamwork for Health and Social Care*. John Wiley & Sons, Ltd, Chapter 4, 57–76. https://doi.org/10.1002/9781444325027.ch4

[7] Steven M. Elias and Scott MacDonald. 2007. Using Past Performance, Proxy Efficacy, and Academic Self-Efficacy to Predict College Performance. *Journal of Applied Social Psychology* 37, 11 (nov 2007), 2518–2531. https://doi.org/10.1111/j.1559-1816.2007.00268.x

[8] Hakan Erdogmus and Cécile Péraire. 2017. Flipping a graduate-level software engineering foundations course. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*. IEEE. https://doi.org/10.1109/icse-seet.2017.20

[9] James B. Fenwick. 2003. Adapting XP to an Academic Environment by Phasing-In Practices. In *Extreme Programming and Agile Methods - XP/Agile Universe 2003*. Springer Berlin Heidelberg, 162–171. https://doi.org/10.1007/978-3-540-45122-8_19

[10] Armando Fox and David Patterson. 2014. *Engineering Software as a Service: An Agile Approach Using Cloud Computing* (1st ed.). Strawberry Canyon LLC, San Francisco, CA.

[11] Alfredo Goldman, Fabio Kon, Paulo JS Silva, and Joseph W Yoder. 2004. Being extreme in the classroom: Experiences teaching XP. *Journal of the Brazilian Computer Society* 10, 2 (2004), 4–20.

[12] Alejandro Guerrero, Rafael Fresno, An Ju, Armando Fox, Pablo Fernandez, Carlos Muller, and Antonio Ruiz-Cortés. 2019. Eagle: a team practices audit framework for agile software development. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 1139–1143.

[13] Nicole Herbert. 2018. Reflections on 17 years of ICT Capstone Project Coordination. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE '18*. ACM Press. https://doi.org/10.1145/3159450.3159584

[14] Martin Hoegl and Hans Georg Gemuenden. 2001. Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization Science* 12, 4 (2001), 435–449.

[15] Steve W. J. Kozlowski. 2017. Enhancing the Effectiveness of Work Groups and Teams: A Reflection. *Perspectives on Psychological Science* 13, 2 (dec 2017), 205–212. https://doi.org/10.1177/1745691617697078

[16] William Krebs. 2002. Turning the Knobs: A Coaching Pattern for XP through Agile Metrics. In *Extreme Programming and Agile Methods — XP/Agile Universe 2002*. Springer Berlin Heidelberg, 60–69. https://doi.org/10.1007/3-540-45672-4_7

[17] Eetu Kupiainen, Mika V. Mäntylä, and Juha Itkonen. 2015. Using metrics in Agile and Lean Software Development – A systematic literature review of industrial studies. *Information and Software Technology* 62 (jun 2015), 143–163. https://doi.org/10.1016/j.infsof.2015.02.005

[18] Yngve Lindsjørn, Dag I.K. Sjøberg, Torgeir Dingsøyr, Gunnar R. Bergersen, and Tore Dybå. 2016. Teamwork quality and project success in software development: A survey of agile development teams. *Journal of Systems and Software* 122 (dec 2016), 274–286. https://doi.org/10.1016/j.jss.2016.09.028

[19] Michelle A. Marks, John E. Mathieu, and Stephen J. Zaccaro. 2001. A Temporally Based Framework and Taxonomy of Team Processes. *The Academy of Management Review* 26, 3 (jul 2001), 356. https://doi.org/10.2307/259182

[20] Maira Marques and Sergio F. Ochoa. 2014. Improving teamwork in students software projects. In *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*. IEEE. https://doi.org/10.1109/cseet.2014.6816787

[21] Zainab Masood, Rashina Hoda, and Kelly Blincoe. 2018. Adapting agile practices in university contexts. *Journal of Systems and Software* 144 (oct 2018), 501–510. https://doi.org/10.1016/j.jss.2018.07.011

[22] Christoph Matthies, Thomas Kowark, Keven Richly, Matthias Uflacker, and Hasso Plattner. 2016. How surveys, tutors, and software help to assess Scrum adoption in a classroom software engineering project. In *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16*. ACM Press. https://doi.org/10.1145/2889160.2889182

[23] Nancy R Mead. 2009. Software engineering education: How far we've come and how far we have to go. *Journal of Systems and Software* 82, 4 (2009), 571–575.

[24] Andreas Meier, Martin Kropp, and Gerald Perellano. 2016. Experience Report of Teaching Agile Collaboration and Values: Agile Software Development in Large Student Teams. In *2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET)*. IEEE. https://doi.org/10.1109/cseet.2016.30

[25] Nils Brede Moe, Torgeir Dingsøyr, and Tore Dybå. 2010. A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology* 52, 5 (may 2010), 480–491. https://doi.org/10.1016/j.infsof.2009.11.004

[26] Matthias M Müller and Walter F Tichy. 2001. Case study: extreme programming in a university environment. In *Proceedings of the 23rd International Conference on Software Engineering. ICSE 2001*. IEEE Computer Society, IEEE Comput. Soc, 537–544. https://doi.org/10.1109/icse.2001.919128

[27] Mary Poppendieck and Tom Poppendieck. 2003. *Lean Software Development: An Agile Toolkit: An Agile Toolkit*. Addison-Wesley.

[28] Pasquale Salza, Paolo Musmarra, and Filomena Ferrucci. 2019. Agile Methodologies in Education: A Review. In *Agile and Lean Concepts for Teaching and Learning*. Springer, 25–45.

[29] Andreas Scharf and Andreas Koch. 2013. Scrum in a software engineering course: An in-depth praxis report. In *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*. IEEE, 159–168.

[30] J.-G. Schneider and Rajesh Vasa. 2006. Agile practices in software development - experiences from student projects. In *Australian Software Engineering Conference (ASWEC'06)*. IEEE. https://doi.org/10.1109/aswec.2006.9

[31] Ken Schwaber and Mike Beedle. 2002. *Agile software development with Scrum*. Vol. 1. Prentice Hall Upper Saddle River.

[32] Todd Sedano, Paul Ralph, and Cecile Peraire. 2017. Software Development Waste. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE. https://doi.org/10.1109/icse.2017.20

[33] Dana E. Sims, Eduardo Salas, and C. Shawn Burke. 2004. Is There a "Big Five" in Teamwork? https://doi.org/10.1037/e518632013-346

[34] Jan-Philipp Steghöfer, Eric Knauss, Emil Alégroth, Imed Hammouda, Håkan Burden, and Morgan Ericsson. 2016. Teaching Agile: addressing the conflict between project delivery and application of Agile methods. In *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16*. ACM Press. https://doi.org/10.1145/2889160.2889181

[35] Willy Wijnands and Alisa Stolze. 2019. Transforming Education with eduScrum. In *Agile and Lean Concepts for Teaching and Learning*. Springer, 95–114.

[36] Sergio Donizetti Zorzo, Leandro de Ponte, and Daniel Lucredio. 2013. Using scrum to teach software engineering: A case study. In *2013 IEEE Frontiers in Education Conference (FIE)*. IEEE. https://doi.org/10.1109/fie.2013.6684866