# Scalable Team-Based Software Engineering Education via Automated Systems

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

Xiao Fu
Harbin Institute of Technology
Harbin, China
fxdawn@hotmail.com

Joshua Zeitsoff
EECS*
University of California, Berkeley
Berkeley, CA, USA
jzeitsoff@berkeley.edu

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

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

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

## I. INTRODUCTION

Team projects are essential in modern software engineering education. Students collaboratively build a piece of software that addresses practical issues, through which they practice both technical skills and soft skills, in a setting that resembles the real working environment. However, team projects are complex. Previous studies have explored various design spaces, such as team formation [1], project selection [2], team coaching [3], and student evaluation [4]; while others have reported experience regarding the design, organization, teaching, and evolution of a project-based software engineering curriculum [5][6][7][8]. The complexity of team projects explains why we rarely see large-scale software engineering courses with a team project. In this paper, we try to address the scalability issue of software engineering projects and lay out our blueprint for enabling the learning of software engineering with a team project via MOOCs.

## II. PROCESSES MATTER

In this section, we explain why software engineering team projects are difficult to scale from the perspective of team coaches; we further show that measuring software engineering processes is the key to scalable software engineering team projects.

We address the scalability issue from coaches' perspective because coaching is necessary to keep team project on track [3], yet the nature of substantial interactions between coaches and student teams makes it a primary impedance for the scalability of team projects [6]. Erdogmus et al. reported based on their experience that the best ratio of coaches and student teams is 1:2 [6], which implies an unbearably large teaching team in a MOOC-scale course to provide effective coaching. Even worse, coaches have to be experts both technically and pedagogically. They should be able to identify issues of student teams and provide a proper amount of interventions to avoid "over-teaching" while providing appropriate scaffolding or taking sufficient orchestration actions [5][3][6].

Digging deeper, we reviewed previous studies on teaching software engineering with team projects with a focus on coaches' responsibilities.

We categorize the most commonly seen responsibilities into three roles:

- An **Expert** answers technical questions, critiques software design, and provides guidance on development processes to keep teams on track.
- A **Grader** gives evaluations to the team and individuals.
- A **Teaching Assistant** handles daily communication, keeps healthy team dynamics, and tracks the learning of individuals, with the goal of ensuring the learning of each student.

Although different in addressed aspects and purposes of interventions, the three roles share one responsibility: being able to evaluate and critique the processes that students have followed to build the product.

Process measurements can support both individual and team assessments [9], help coaches to better address process issues, and facilitate the tracking of learning issues. Thus our approach to address the scalability of team projects focuses on measuring processes reliably and automatically, and how can we leverage the measurements to inform instructors and students. We will present in the next section more details about this approach.

## III. OUR APPROACH

### A. Systems

Our approach focuses on building automated systems to facilitate coaching of software engineering teams. Our systems address two issues

- How can we effectively give informative feedback at large scale?

*Department of Electrical Engineering and Computer Sciences

- How can we scaffold processes to reduce the need for team coaching?

Our systems are composed of metrics built from *teamwork telemetry*, data available from various development tools. It is a trend in software engineering courses to use tools to support development. Many researchers and instructors have recognized the importance of using tools in software engineering courses with a team project [5][8]. Tools such as the code-hosting and team-collaboration tool GitHub[1], the project-management tool Pivotal Tracker[2], and the code quality measurement service CodeClimate[3], give powerful infrastructure for team development, and in the meantime records behaviors of the software development process. We call data available from these tools *teamwork telemetry* and build process metrics based on the data.

Teamwork telemetry, together with metrics built on top of it, works as building blocks for automated systems. We propose to improve the scalability of team coaching with two systems:

- An early-warning system.
- A scaffolding system.

*a) The early-warning system:* addresses the scalability issue by reducing the workload of instructors to provide informative feedback. Composed of process metrics, the system can provide warnings at an early stage; instructors can thus be aware of teams that need help and provide interventions with information from the system. The system addresses two major issues:

- it reduces the workload of instructors with a dashboard for team status;
- it facilitates less experienced teachers to conduct expert diagnosis and give professional feedback.

*b) The scaffolding system:* improves scalability by reducing the team's need for human coaching. The system gives instructions on the team's tasks and processes. Furthermore, the system is able to track a team's task status and progress. Team members can use the information to improve their practice and navigate through the project.

Figure 1 shows how teamwork telemetry, metrics, and the two automated systems are related.

### B. Role of Instructors/Coaches

Although our systems are designed to address team projects' intensive dependence on the teaching staff, they are not meant to replace instructors. Instead, the systems facilitate instructors by presenting summarized results that used to require a huge amount of effort for instructors to elicit; instructors' responsibilities are pivoted toward decision making and intervention design.

This idea follows Baker's proposal of "stupid tutoring system" [10], where the intellectual power of human is respected and put in the center, while the computational power of machines is used to support human decision making.

[1]https://github.com/
[2]https://www.pivotaltracker.com/
[3]https://codeclimate.com/

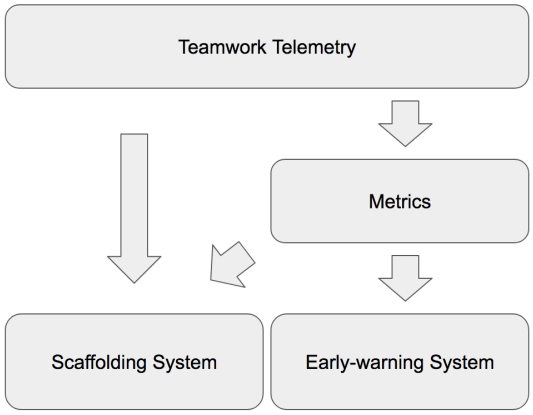


Fig. 1. Both the early-warning system and the scaffolding system are built on top of teamwork telemetry. Metrics are primarily used in the early-warning system, but are also used in the scaffolding system to test the quality of a finished task.

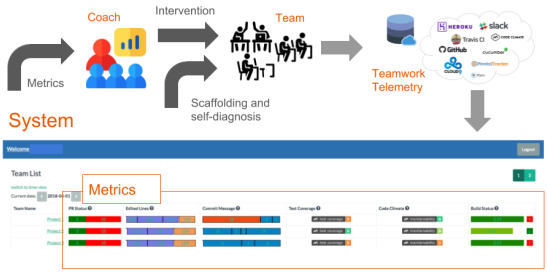


Fig. 2. Our system provides metrics for instructors to facilitate coaching. Students can also directly benefit from the system through scaffolding and self-diagnosis.

Figure 2 shows how instructors and student teams interact with the systems. In a MOOC setting, students can receive self-diagnosis and scaffolding directly from the system, which reduces the workload of instructors. In the meantime, instructors can leverage process metrics to detect abnormalities in student teams and provide targeted feedback efficiently when a team needs help.

### C. Metrics

Previous studies have explored the use of metrics in teaching software engineering courses [11][12]. Matthies et al. further summarized their metrics into a system called ScrumLint [13].

However, their metrics are empirical, designed based on researchers' experiences and observations, and sometimes these empirical metrics may even seemingly conflict with each other. For example, both Alperowitz et al. and Matthies et al. proposed metrics to measure the time to close a pull request on GitHub; while Alperowitz et al. argue that a pull request should be quickly reviewed and closed [11], Matthies et al. think a short close time indicates a violation of the reviewing process [12]. This seeming conflict is induced by

the fact that the two group of researchers address different concerns with this metric, while neither has a thorough understanding of the role of this process.

We ground our metrics on industry practices. Our metrics are designed based on industry's recognized practices, industrial case studies, observations of industry projects, and interviews with experienced practitioners. As for approach, we build a *conformance template* for each process [14]. Conformance templates give us the flexibility when defining a process and allows us to focus on violations of processes, which is helpful in classroom settings.

### D. Summary

Results from a case study in a software engineering course at a US university [15] have provided supporting evidence to our approach by showing that

- measuring processes can provide useful information for team coaches, and
- processes can be measured directly or indirectly from teamwork telemetries.

As our next step, we plan to study more processes and measurements, implement our scaffolding system and early-warning system, and deploy systems in real large-scale software engineering courses.

In summary, with our metrics and the system, a coach can manage multiple teams at the same time, which improves the scalability of software engineering projects. Furthermore, the system pivots coaches attention from the ocean of artifacts and logs to aggregated results mined from teamwork telemetries; thus a less experienced coach can still give accurate feedbacks to student teams.

Our system and metrics can be used as building blocks for more powerful tools to facilitate learning in large-scale software engineering projects. We expect to build an ecosystem that better connects software engineering education with industrial needs. Such an ecosystem will enable the execution of a full project life-cycle, team formation, project selection, customer contact, team management and evaluation, project maintenance, etc., with minimal human inputs.

## ACKNOWLEDGMENT

This research has been partially funded by MOE Online Education Research Center (Quantong Fund) grant 2017ZD203.

This research has also been partially funded by the European Union (grant agreements no. 669074 and 731685), the Spanish Ministries of Economy and Competitiveness (projects TIN2014-53199-C3-2-R and TIN2017-85179-C3-2-R) and Science and Education (PRX17/00410), and the Regional Government of Castilla y Len (project VA082U16).

## REFERENCES

[1] A. Mujkanovic and A. Bollin, "Improving learning outcomes through systematic group reformation-the role of skills and personality in software engineering education," in *Cooperative and Human Aspects of Software Engineering (CHASE), 2016 IEEE/ACM*. IEEE, 2016, pp. 97–103.

[2] T. Sedano, A. Rengasamy, and C. Péraire, "Green-lighting proposals for software engineering team-based project courses," in *Software Engineering Education and Training (CSEET), 2016 IEEE 29th International Conference on*. IEEE, 2016, pp. 175–183.

[3] G. Rodríguez, Á. Soria, and M. Campo, "Measuring the impact of agile coaching on students performance," *IEEE Transactions on Education*, vol. 59, no. 3, pp. 202–209, 2016.

[4] H. Igaki, N. Fukuyasu, S. Saiki, S. Matsumoto, and S. Kusumoto, "Quantitative assessment with using ticket driven development for teaching scrum framework," in *Companion Proceedings of the 36th International Conference on Software Engineering*. ACM, 2014, pp. 372–381.

[5] A. Scharf and A. Koch, "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, 2013, pp. 159–168.

[6] H. Erdogmus and C. Péraire, "Flipping a graduate-level software engineering foundations course," in *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track*. IEEE Press, 2017, pp. 23–32.

[7] N. Herbert, "Reflections on 17 years of ict capstone project coordination: Effective strategies for managing clients, teams and assessment," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 2018, pp. 215–220.

[8] D. Delgado, A. Velasco, J. Aponte, and A. Marcus, "Evolving a project-based software engineering course: A case study," in *Software Engineering Education and Training (CSEE&T), 2017 IEEE 30th Conference on*. IEEE, 2017, pp. 77–86.

[9] F. Rocha and E. Stroulia, "Understanding individual contribution and collaboration in student software teams," in *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*. IEEE, 2013, pp. 51–60.

[10] R. S. Baker, "Stupid tutoring systems, intelligent humans," *International Journal of Artificial Intelligence in Education*, vol. 26, no. 2, pp. 600–614, 2016.

[11] L. Alperowitz, D. Dzvonyar, and B. Bruegge, "Metrics in agile project courses," in *Software Engineering Companion (ICSE-C), IEEE/ACM International Conference on*. IEEE, 2016, pp. 323–326.

[12] C. Matthies, T. Kowark, M. Uflacker, and H. Plattner, "Agile metrics for a university software engineering course," in *Frontiers in Education Conference (FIE), 2016 IEEE*. IEEE, 2016, pp. 1–5.

[13] C. Matthies, T. Kowark, K. Richly, M. Uflacker, and H. Plattner, "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*. ACM, 2016, pp. 313–322.

[14] N. Zazworka, K. Stapel, E. Knauss, F. Shull, V. R. Basili, and K. Schneider, "Are developers complying with the process: an xp study," in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. ACM, 2010, p. 14.

[15] A. Ju and A. Fox, "Teamscope: measuring software engineering processes with teamwork telemetry," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 2018, pp. 123–128.