The DevOps Lab Platform for Managing Diversified Projects in Educating Agile Software Engineering

Xiaoying Bai, Dan Pei, Mingjie Li, Shanshan Li
Department of Computer Science and Technology
Tsinghua University, Beijing, China, 10008
Email: {baixy, peidan, lishanshan}@tsinghua.edu.cn, li-mj14@mails.tsinghua.edu.cn

***Abstract*—This Research Work-in-Progress paper presents the design of a Software Engineering (SE) course to support project-based practical training. Group projects, especially projects from industry partners, are deemed to be necessary for students to gain hands-on experiences. With projects from the real world, students learn not only practical engineering solutions, but also the context, constraints, and social aspects of SE. For a course having over 100 students with different interests and experiences, it is desired to provide diversified choices of projects to stimulate enthusiasm for learning. However, management and evaluation of diversified projects are challenging. Following the Agile principles, we need to continuously track progress and activities of each group, to provide quick feedback of deliveries, and to periodically evaluate students' performance. Therefore, we built a DevOps platform based on GitLab version control and continuous integration framework. Commits to GitLab code repositories automatically trigger build, testing, and analysis functions (which provide both qualitative and quantitative feedback to the students). This system has been in operations since 2014 for an undergraduate SE course, with over 500 students participating in over 130 project teams in total. The preliminary research showed promising results in improving SE education.**

I. Introduction

Project-based practical training has been widely accepted as an important part of Software Engineering (SE) education [1] [2] [3]. With carefully designed projects, especially projects from industry partners, students learn not only practical engineering solutions, but also the context, constraints, and social aspects of SE [4] [5]. They gain hands-on experiences of software lifecycle process, development and management techniques, communication and collaboration skills.

For SE courses in general, students are usually assigned with a unified project with well-defined requirements. However, for courses having over 100 enrollments in a semester, one fixed project for all students could be boring. A reasonable solution is to introduce diversities into the design of course projects. On one hand, it provides diversified choices so that a student has the flexibility to choose one that best fits his/her background and interests. On the other hand, it can simulate uncertainties in real project context as students face different customers, requirements, and development environment. The variety of software projects is also helpful to enrich the students' experiences and stimulate cross-project knowledge sharing. Therefore, we collaborate with various partners representing customers from industry as well as on-campus organizations, and design projects representing different customer types, architecture styles and techniques. Some example projects are like VOD (Video on Demand) system for students' broadcasting association, class sign-in system based on face-recognition using cognition Web APIs from open platforms, and vehicle entertainment system for Automobile Research Institute.

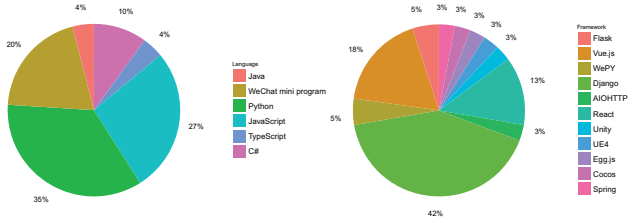
While inspiring learning interests, such diversity also presents research challenges to process supervision and quality assessment, which existing software engineering platforms (for experienced software engineers) cannot directly deal with. It is very challenging to fairly and timely evaluate and provide feedback to students who have diversified customers, diversified development environments and techniques, diversified experience, and diversified roles in the project team, while the instructor-to-student ratio is low (*e.g.*, about 1:100 in our case) and the SE project has to compete with several other heavy projects for a student's time.

The goal of our platform's is to have a unified platform that can address above challenges. We devised a customized Agile process to encourage both online and offline communication and collaboration among team members, customer representatives, and teaching assistants who take the role of product manager. With limited resources of lecturers and teaching assistants, it may take a lot of time and efforts to supervise every project and every student. Automatic tools are thus demanded to facilitate project data collection and analysis [1] [6] [7] [8]. Taking DevOps practices, we built a collaborative development platform by integrating GitLab with plug-in tools for continuous integration, testing, deployment, and assessment. We designed various automatically generated reports to provide timely feedback to students, customers, and TAs. The platform was incrementally constructed and applied during the last four years. It has a total of over 500 students participating in over 130 project teams. Preliminary results showed promising improvements in training students' development habits and strengthening process quality control.

II. Course and Project Design

The research is for the course of "Introduction to Software Engineering", which is a mandatory course for all undergraduate students in the Department of Computer Science and Technology in Tsinghua University. There are about 150 enrollments each semester, coming from 2^{nd} and 3^{rd} year students. Students have background in programming languages

978-1-5386-1174-6/18/$31.00 ©2018 IEEE



(a) Diversified languages (b) Diversified frameworks

Fig. 1. An example of project diversity. The data are collected from the 40 projects of 2017 Fall semester.

and data structures, but limited experiences in complex system design and development, projects management, and team collaboration. For most of the students, the average size of the largest software they have ever built is less than 1,000 LOC (Lines of Code). SE course thus aims to train students from following perspectives:

- End-to-End system architecture design for a relative complicated problem.
- Team-based collaborative development with 3-5 students in each team.
- Project planning, task decomposition and scheduling.
- Engineering habits training to follow good development practices such as eliminating bad code smells [9], continuous version control, and coverage-based testing.

Many studies have illustrated the benefits of introducing real industry projects to SE course education. But, as Hans pointed out [10], considering the students' background and the limitations of course syllabus, it could be very hard to find industry projects with proper size, complexity and workload. To deal with this problem, the industry projects selected were customized with following constraints: the team size (3-5 students each team), time (10 weeks during a semester), students' background, and course syllabus.

A. Project Diversity

Diversity is a principle when we designed the course projects. Software have been widely used in all aspects of industrial production and social life. Different types of software require different SE methods, techniques, tools, and even the social aspects of software development. As a fundamental and compulsory course for students majoring in computer-related specialities, we hope that students can experience modern development technologies and best practices as much as possible in the course-project environment. We thus selected various topics and allowed students to choose different techniques such as programming languages, Web frameworks, xUit tools, third-party components/services, and open source packages. Requirements were divided into *mandatory*, *optional*, and *creative* parts. While mandatory and optional requirements were predefined, creative requirements encourage students to creatively design their own features. Taking the course of 2017 Fall semester as an example, Figure 1 shows the diversified languages and frameworks used in the 40 projects.

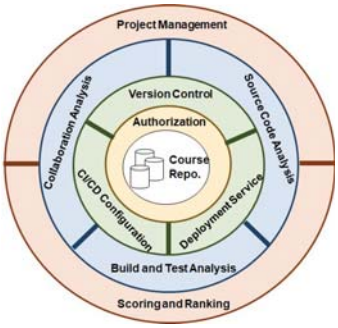


Fig. 2. The layered architecture of the proposed framework, centered around the repository of course data. The infrastructure services provide authorization and runtime environment to support CI/CD. The data are collected throughout the process for analysis and assessment.

B. Agile Process

The Agile principles have been successfully applied to industry software system constructions, which promote frequent small and incremental deliveries, test-driven development, and adaptability to changes. College courses are also reformed to accommodate the paradigm shift in modern software industry [1] [3] [11] [12].

In this research, we customized Agile Scrum process to fit the process model into the semester time schedule and course education requirements. The process was divided into 5 iterations with 1-2 weeks for each Scrum Sprint. At each iteration, students met with customer representatives. They reviewed the phased progress of delivered functionalities, and agreed on the work plan of next iteration. During the iterative and incremental development process, students were expected to progressively improve SE skills with accumulated domain knowledge and development experiences, such as requirements acquisition and analysis, to make tradeoffs of functional and non-functional requirements, to break down requirements into manageable small tasks, to decompose system into decoupled modules, and to react to changes in requirements and design.

In spite of project diversity, we defined the uniform format of process and project management, including project planning and task allocation, source code version control, unit testing, Cloud-based software delivery and deployment. With tool support, the engineering practices were continuously re-enforced throughout the project lifecycle.

III. The DevOps Framework

DevOps supports Agile practices with an infrastructure of automatic tool chain to support continuous integration, testing, quality control, and deployment [7].

A. The Platform

In this research, we built a DevOps platform based on GitLab[1] code repository and continuous integration framework, and hosted it on the Cloud platform. Figure 2 shows the layered architecture of the proposed framework.

[1]https://www.gitlab.com

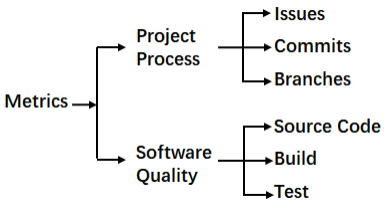


Fig. 3. The metrics defined for evaluating project process and software quality from different perspectives, which are calculated for individuals, teams, projects, and repositories, at various frequencies and on demand.

- The repositories of course data, which include code, project and process data, analysis and assessments.
- The authorization mechanism, which defines access privileges of various roles for each type of course data, such as students, teaching assistants, customer representatives and instructors.
- Runtime environment, which provides the infrastructure services to support CI/CD (Continuous Integration/Continuous Deployment) including version control, deployment services and CI/CD configurations.
- Process analysis, which is exercised from three perspectives including:
  - Source code analysis, such as the code smells [9], coding styles, vulnerabilities, and maintainability.
  - Build and test results analysis, such as test coverage and bug reports.
  - Collaboration analysis, such as task allocation, scheduling, team member contribution, frequency and adequacy of commitments.
- Project assessment, which defines various metrics to quantitatively evaluate each project, each team, and each student.The final grade also takes into consideration the offline assessment from teaching assistants and customer representatives based on their interactions with project teams.

B. Continuous Assessment

Quality is assessed with following principles:

- To evaluate not only team achievements, but also individual contributions;
- To evaluate not only final deliveries, but also progress performance; and
- To evaluate not only software development, but also project management.

Various metrics are defined to provide a quantitative view of project progress, status, and quality, as shown in Figure 3. The data are collected from source code repository, version control system, build and test reports, and code analysis tools. They are calculated for teams as well as individuals. Tools are built to automatically generate reports at various frequencies or on demand.

We encourage students to use GitLab issue management for project task management, and use issue statistics to monitor project progress. Issues are considered as tasks annotated with types, status, and priorities. Issue board is used to facilitate task planning and organization, and to visualize progress by tracing issues status. Statistics include the number of issues opened at the beginning of each iteration, the percentage of closed issues at the end of an iteration, the time duration of each issue, the issues allocated to each team member, and the fluctuations of issue statistics (e.g., total number and distribution among team members) per iteration throughout the lifecycle. Students are expected to gradually improve the skills of balancing tasks among team members, realistic workload estimations, and iterative deliveries on schedule.

Commits are used to supervise students' version management and collaboration skills. For students who are new to collaborative development, we observed some common problems of commits. For examples, beginners tend to push large modifications and submit with insufficient comments. In the course project, we designed the metrics to guide students to form the habits of commit with small and frequent modifications, and concrete and specific commit messages. Code size, message length, and commit frequency are thus taken as three indicators. Students will get warnings if the system detects abnormal commits such as over-size, over-simplified comment messages, over high/low frequency, and so on.

Branching is widely used in parallel development. There exist various branching patterns in industry environment which are largely dependent on software and project types, and organization culture. However, in the course exercise for beginners, we intend to train students the following practices [13] [8]:

- Merge Your Own Code. One can only merge his or her own code to enforce students taking responsibilities of the code.
- Early Branching. Independent tasks are suggested to be divided into different branches to parallelize the development.
- Merge Often. This is to encourage small incremental deliveries in each iteration and train students' capabilities of resolving conflicts.

Every commit triggers the events of source code quality control by integrating static analysis tools and testing frameworks. Many tools are available to diagnose bad smells (any characteristic in the source code of a program that possibly indicates a deeper problem [9]) in the code such as coding style, code comments, structure complexity, code clone, violation of safety rules, and security vulnerabilities. Integration the reports from the tools can reinforce students the engineering habits. In addition to static analysis, we also require students to submit with testing scripts, including unit, functional, and performance testing. By configuring the CI pipeline process, the software are deployed in a container environment, test scripts are automatic executed, bug and performance reports are generated and collected for analysis.

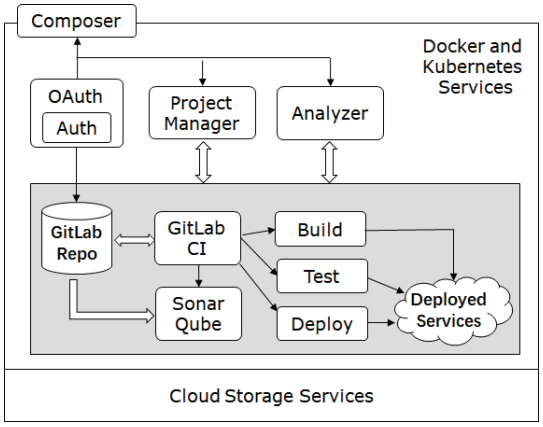
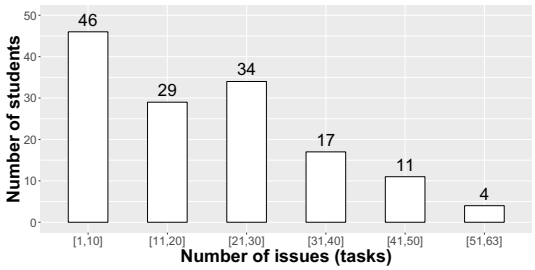
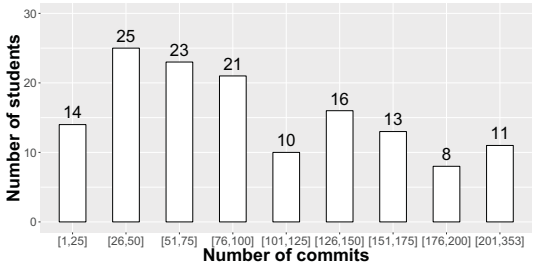


Fig. 4. The prototype system built on Cloud platform. Taking Kubernetes micro-service architecture, the system is composed of multiple services including project management, GitLab repository, Auth authorization, GitLab CI, SonalQube code analysis, deployment, build, test, and analysis.



(a) Issues by Individual



(b) Commits by Individual

Fig. 5. Statistics of the number of issues and commits by individual students.

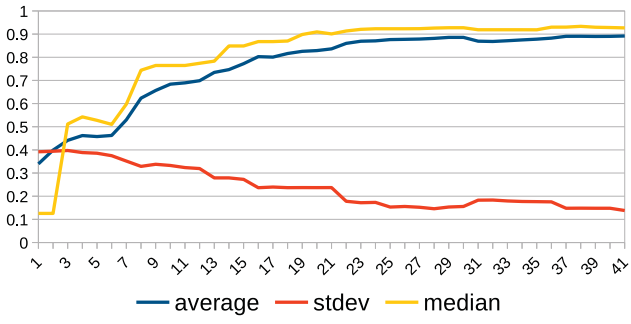


Fig. 6. COMMIT assessment for 2016 course. In consecutive 41 days, the average and median increase while standard deviation decreases progressively.

IV. Prototype and Results

A prototype system was built on Cloud platform. Figure 4 shows the micro-service architecture of the prototype system. The components in Figure 2 are wrapped and deployed as Kubernetes[2]/docker[3] containerized services.

In the 2017 Fall semester, there are 141 students enrolled in the course, organized into 33 teams participating 11 projects. Altogether 68 code repositories were created in GitLab, with 3076 issues and 14506 commits throughout the 64 days of project development. Figure 5 shows the statistics of total number of issues (Figure 5(a)) and commits (Figure 5(b)) by individual students. On average, over 70% students were assigned at least 2 issues at each iteration, and committed at least once per day; over 30% students committed at least twice per day. Those exceptional inactive students got warnings to keep up with teammates. Students with many tasks and commits also got attentions from instructors to avoid overload or improper operations.

Students' Improvement. With frequent feedback and assessment, we can see the improvement in students' engineering practices *progressively*. For example, Figure 6 shows, in a period of 41 days in Fall 2016, the average and median $COMMIT$ gradually increases, and the standard deviation gradually decreases. The results indicate that, as expected, with the help of our platform, students became more and more accustomed to and better at good engineering habits. As a qualitative comparison, another SE course in the same university, which did not use our platform but collected the stats similar to $COMMIT$, did not see a similar trend clearly.

The system has been refactored and upgraded after each semester course in the last four years, until this year we built the tool chain covering full lifecycle process. The data were gradually accumulated. Some data showed clear improvements. More data will be collected and analyzed in the future.

[2]https://kubernetes.io/
[3]https://www.docker.com

V. Conclusion

We propose to enhance the openness and diversity of project-based SE courses through a well-designed framework of project process supervision and assessment. The paper reported our research in recent 4 years to innovate course design based on the Agile best practices, with DevOps tools supporting the pipeline process of continuous integration and quality control. The preliminary results showed promising improvements in education and practical training, which we believe are good reference for SE courses in general.

VI. Acknowledgment

The authors would like to thank all our project partners, the instructors and students who support the course development. Special thanks to Dr. Wolfgang Mauerer and his team for sharing with us Codeface tools, their knowledge and experiences.

REFERENCES

[1] P. A. Laplante, "An agile, graduate, software studio course," *IEEE Transactions on Education*, vol. 49, no. 4, Nov 2006. [Online]. Available: http://dx.doi.org/10.1109/TE.2006.879790

[2] "Enriching traditional software engineering curricula with software project management knowledge," 2016. [Online]. Available: http://dx.doi.org/10.1145/2889160.2889193

[3] R. Chatley and T. Field, "Lean learning - applying lean techniques to improve software engineering education," *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, May 2017. [Online]. Available: http://dx.doi.org/10.1109/ICSE-SEET.2017.5

[4] I. Liem, Y. Asnar, S. Akbar, A. Mulyanto, and Y. Widyani, "Reshaping software engineering education towards 2020 engineers," in *Proceedings of the IEEE CSEE&T 27th Conference on Software Engineering Education and Training - CSEE&T '12*. IEEE Press, 2014. [Online]. Available: https://ieeexplore.ieee.org/document/6816797/

[5] M. R. Marques, "Monitoring: An intervention to improve team results in software engineering education," in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, ser. SIGCSE '16. New York, NY, USA: ACM, 2016. [Online]. Available: http://doi.acm.org/10.1145/2839509.2851054

[6] L. Alperowitz, D. Dzvonyar, and B. Bruegge, "Metrics in agile project courses," 2016. [Online]. Available: http://dx.doi.org/10.1145/2889160.2889183

[7] J. Feliciano, M.-A. Storey, and A. Zagalsky, "Student experiences using github in software engineering courses," 2016. [Online]. Available: http://dx.doi.org/10.1145/2889160.2889195

[8] R. B. Rayana, S. Killian, N. Trangez, and A. Calmettes, "Gitwaterflow: a successful branching model and tooling, for achieving continuous delivery with multiple version branches," *Proceedings of the 4th International Workshop on Release Engineering - RELENG 2016*, 2016. [Online]. Available: http://dx.doi.org/10.1145/2993274.2993277

[9] M. Fowler, *Refactoring. Improving the Design of Existing Code*. Addison-Wesley, 1999, no. ISBN 0-201-48567-2.

[10] H. van Vliet, "Reflections on software engineering education," in *Inverardi P., Jazayeri M. (eds) Software Engineering Education in the Modern Age. ICSE 2005. Lecture Notes in Computer Science, vol 4309. '12*. Springer, Berlin, Heidelberg, 2006. [Online]. Available: https://doi.org/10.1007/11949374_1

[11] C. Anslow and F. Maurer, "An experience report at teaching a group based agile software development project course," in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '15. New York, NY, USA: ACM, 2015, pp. 500–505. [Online]. Available: http://doi.acm.org/10.1145/2676723.2677284

[12] M. Paasivaara, J. Vanhanen, V. T. Heikkila, C. Lassenius, J. Itkonen, and E. Laukkanen, "Do high and low performing student teams use scrum differently in capstone projects?" *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)*, May 2017. [Online]. Available: http://dx.doi.org/10.1109/ICSE-SEET.2017.22

[13] B. Appleton, S. P. Berczuk, R. Cabrera, and R. Orenstein, "Streamed lines: Branching patterns for parallel software development," *PLoP*, 1998.