Outcome-based School-Enterprise Cooperative Software Engineering Training*

Kun Ma†
School of information science and engineering, University of Jinan
Jinan 250022, Shandong
ise_mak@ujn.edu.cn

Bo Yang
School of information science and engineering, Linyi University
Linyi 276000, Shandong
yangbo@ujn.edu.cn

Jin Zhou
School of information science and engineering, University of Jinan
Jinan 250022, Shandong
ise_zhouj@ujn.edu.cn

Yongzheng Lin
School of information science and engineering, University of Jinan
Jinan 250022, Shandong
ise_linyz@ujn.edu.cn

Kun Zhang
School of information science and engineering, University of Jinan
Jinan 250022, Shandong
ise_zhangk@ujn.edu.cn

Ziqiang Yu
School of information science and engineering, University of Jinan
Jinan 250022, Shandong
ise_yuzq@ujn.edu.cn

ABSTRACT

On the background of Emerging Engineering Education (3E) in China, it has been an important topic in school-enterprise collaboration in software engineering education. In this paper, our outcome-based school-enterprise cooperative software engineering training is proposed to increase student engagement of engineering practice, and address some real complicated engineering issue. Contributions of our method are project-driven agile practice using pair programming and extreme programming, school-enterprise collaboration to address complicated engineering issues, and outcome-based encouragement to formulate a case library. Finally, complete, time estimation and quality of work order, and detailed survey and feedbacks of students are analyzed to illustrate the effect of our software engineering training.

CCS CONCEPTS

• **Applied computing** → *Collaborative learning*;

KEYWORDS

Software Engineering, School-Enterprise Cooperation, Case Library

ACM Reference Format:
Kun Ma, Bo Yang, Jin Zhou, Yongzheng Lin, Kun Zhang, and Ziqiang Yu. 2018. Outcome-based School-Enterprise Cooperative Software Engineering Training. In *TURC 2018: ACM Turing Celebration Conference - China, May 19–20, 2018, Shanghai, China.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3210713.3210722

*Produces the permission block, and copyright information
†Corresponding author.

TURC 2018, May 19–20, 2018, Shanghai, China

ACM ISBN 978-1-4503-6415-7/18/05…$15.00
https://doi.org/10.1145/3210713.3210722

1 INTRODUCTION

School-Enterprise Collaboration in Software Engineering has been an important topic in a technology-enabled connected world. Industrial impact of research is of utmost importance in the area of engineering. For instance, ACM SIGSOFT has proposed Impact project (www.sigsoft.org/impact) to measure and analyze the impact of software engineering research in practice. Various workshops and panels are regularly organized during the construction of Emerging Engineering Education (3E) in China [9].

The global software industry and academia are two large communities. However, unfortunately, the level of joint industry-academia collaborations in school-enterprise cooperative software engineering training is still relatively very low, compared to the amount of activity in each of the two communities [5] [7]. Previous study gave some incorrect results in software engineering experiments [6]. Research practices must improve to increase the trustworthiness of software engineering experiments. A key to this improvement is to avoid conducting studies with unsatisfactory low statistical power.

In this paper, we propose outcome-based school-enterprise cooperative software engineering training. Outcome is the training objective to cultivate students both personal ability and teamwork. We design online training, lectures and team project to promote the communications between students and teachers. Contributions of our method are project-driven agile practice using pair programming and extreme programming, school-enterprise collaboration to address complicated engineering issues, and outcome-based encouragement to formulate a case library.

The remainder of the paper is organized as follows. The related work is discussed in Section 2. In Section 3, we introduce our software engineering training objective. Section 4 presents the training design including online training, lectures, and team project. In Section 5, the training of agile

development process is divided into project startup, requirement and iterative development. we evaluate our experience in several dimensions, such as complete, time estimation and quality of work order, survey of student engagement, and feedback from students. Brief conclusions are outlined in the last section.

2 RELATED WORK

Teachers in some universities teach software engineering with object-oriented waterfall-like model [10] [11]. The development process is divided into requirement, design, implementation and test. This method ignores the changes of development techniques in reality. It leads to some inconsistency between the knowledge architecture and market talent need.

There are also some new ways to teach software engineering. For example, Massive Open Online Course (MOOC) is used everywhere to teach students online [13]. However, this way has some drawbacks on training practice due to the lack of face-to-face interactions between teachers and students. Another method is case teaching for engineering education [8]. Student can repeat some experiments of the case library. This method is effective if the successful cases can be repeated. But it sometimes kills innovation [16]. Another method is aspect oriented programming for software engineering training [2]. They experience software engineering training in academic and industrial settings. These principles are independent of the technologies to which they must be applied and of the measurement and load generation tools that support them. Another study shows software engineering curricula can be enhanced by incorporating SWEBOK knowledge and standards [1].

Learn from doing is the basic principle to achieve even higher quality results. Teachers in Microsoft attempt to take the development of an interactive game as a typical case with a strategy for teaching university students the dynamics of a software project [15]. Teachers in Jackson State University taught software engineering through the use of innovative mobile application development [14]. Teachers in University of Cdiz introduced the administration tool of the serious game ProDec, that allow trainers to design the game scenarios of the game trying to overcome the lacks found in the scope of serious games for Software Project Management (SPM) [3]. Another method is crowdsourcing in software engineering [12]. Industrial crowdsourcing practice is used in software engineering training.

To bridge the gap between industry and academia and to foster software engineering training, a number of researchers from academia and also practitioners from industry have systematically studied and reported challenges and best practices. Systematic Literature Review and systematic mapping process are analyzed [5].

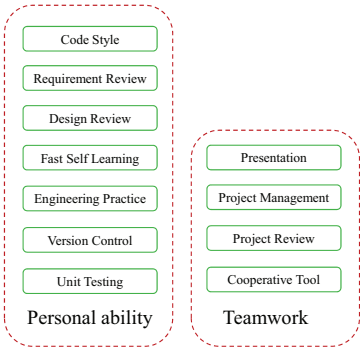

Figure 1: Training objectives.

3 TRAINING OBJECTIVE

Objectives of school-enterprise cooperative software engineering training are divided into personal ability (online cultivation) and teamwork (face-to-face collaboration). Figure 1 shows the training objectives.

Personal ability means research and development ability to analyze, design, and implement a software system. First, we wanted instructors and pupils to learn normalized and generalized code style. We select Alibaba Code Style as the reference, which is influential in software industry in China. Second, we wanted students to experience requirement review in practice. All the subjects of software project are from the actual needs of enterprises. Third, we wanted students to lean how to review requirement and design to solve complicated engineering problems. Forth, we wanted students to have fast self learning ability to learn necessary knowledge. Fifth, we wanted students to experience the strenuous process to meet and address issues in engineering practice. Last, we wanted students to master some development tools for skills of unit testing, version control and daily build.

Teamwork means the ability to organize a team for development collaboration. First, we wanted students to express ideas, solutions, and software works. Second, we wanted students to experience agile development process including project startup and iterative development. Last, we wanted students to use software collaborative tools to manages source codes and software revisions.

4 TRAINING DESIGN

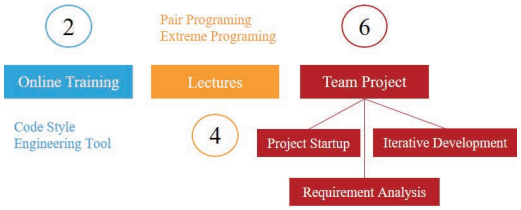

Figure 2: 3 key stages.

We wanted students to understand the principles of software engineering and experience normalized agile development process. Therefore, we designed 3 key stages shown as Fig. 2 to learn software engineering, and made our assignments to be sufficiently measurable. We made some testing scripts to evaluate the performance of the systems students develop, so that each assignment is markable.

The first stage is 2 weeks of online training to cultivate personal development ability, such as code style and engineering tools. The second stage is 4 weeks of lectures to teach software engineering knowledge and present enterprise practical cases. The third stage is 6 weeks of development practice to solve complicated engineering problems. Most of cases come from the requirement of the enterprise. The enterprise provides us some technological innovation funding.

Some feedbacks of students indicate that students cannot learn much using waterfall-like process model as we expect. Therefore, we only adopt agile development process model for a team project. Several students (from 5 to 7) are grouped to develop the applications together. The process of a team agile project is divided into requirement, project startup and iterative development. We have an intermediate check in the middle of the development of the team project, and a final check in the end.

4.1 Online Training

We made a survey on the students who select our course in the beginning. Most of them have learn at least 1 object-oriented programming language, and have learn data structure and database. Originally, our programming language of the assignments was Java. This relied on that Java course is taught before the fifth semester in our university.

In the design of online training, we introduce the code style of Alibaba, and the usage of some engineering tools such as SVN (version control), JUnit (unit testing), ANT (daily build), and Axure (rapid prototyping design). To make students learn the occupation of software engineer, we also teach students what accomplishments a software engineer MUST have.

4.2 Lectures

Beside online training, we had 4 weeks of lectures (average 2 class hours per week). The contents of lectures include pair programming and extreme programming. Besides, some enterprise technical solutions are introduced in this stage.

4.3 Team Project

We provide students a case library as a reference to determine the topics of the project. This library includes several successful historical cases that come from innovative practices.

The enterprise gives the actual requirements, and provides funding for software solutions. Students can compete against each other to apply for the project, then school-enterprise reviews all the proposals to determine which solution is more practical. After the completion of the project, school-enterprise checks the result and offers cash prizes. At the end of the team project, we hold a conference to honor some excellent groups.

5 AGILE DEVELOPMENT PROCESS

5.1 Project Startup

First, each development group is to determine the content of the team project. There are 10 cases in our case library, which come from different industry. We call this Internet Plus cases. The members of each development group are volunteered to come together. We just control the gender and competence balance of all groups to make more average.

5.2 Requirement

Next, we organize all groups to make preliminary requirement analysis with role-based method [4]. For the topics from the enterprise, groups of students need have discussion with their actual clients. For the topics from the case library, the requirements are from the prototype system. For the topics from the teachers, the requirements are from the discussion of teachers. For the user-defined topics, the requirements are from the markets.

User story cards and mind map are used to conclude requirements. At the end of project startup, we organize several clients and enterprise software engineers to checks all the project deliverables including software prototype, software architecture, business process, development schedule, and task decomposition. In order to make students understand what real requirements are, we invite 3 software engineers from different enterprises to present 3 reports about requirement analysis methods.

5.3 Iterative Development

After project startup, groups of students attempt to design more detailed solution of software project using agile development thoughts. Extreme programming and pair programming is used to track the project schedule. Periodical software distribution makes all groups anxious to succeed better.

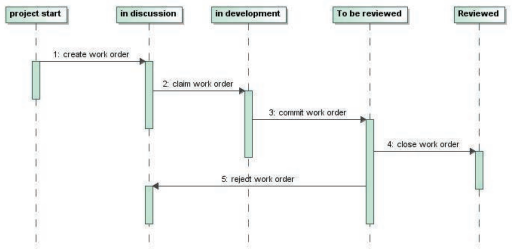

Figure 3: Sequence diagram of work orders.

Iterative development is driven by work orders. Project manager is responsible to create work orders by the development plan. Then, the work order is circulated. Fig. 3 shows the sequence diagram of work orders. All the executive state of work orders is put online to make students more

Table 1: Checking list of work order.

No.	Check Item
1	A work order is corresponding to one and only one branch.
2	The comment of each commit is accurate or not.
3	The comment of each commit contains the number of work order or not.
4	The comment of each commit contains the time cost of work order or not.
5	The open and close of work order is paired or not.
6	The work order is completed on time or not.

competitive. The work orders are classified by the development milestone. When the development work of work order is complete, the source codes of branches are pushed into the trunk. We provide a checking list of code commit shown in Table 1 for peer evaluations. Each commit is checked by the pre-commit hook script we develop. At the end of iterative development, commits are checked manually by teachers at the same time.

During iterative development, the document of software engineering is also checked. We organize several cross tests of other project groups. When a new bug is checked out, a new work order to solve this bug is made. During cross tests, a score of self group is added when finding bugs of other groups. A score of self group is minus several points when a bug of self group is found by other groups. Strong competition engaged all students defending together. At the end of iterative development, the demonstration of software works is shown. Each member of a group has the opportunity to present the work of this group.

6 PERFORMANCE EVALUATION

All students experience 6 weeks' practice training of a team project. Generally, our outcome-based school-enterprise cooperative software engineering training worked well. Even so, what is more worthy of reflection are several lessons learned from data statistics and engineering practice.

6.1 Complete of Work Order

Fig. 4 shows the percentage complete of work order of each project group. After 6 weeks' normalized practice, all groups are used to finishing work order on time. As shown in Fig. 4, the average percentage complete of work order increases rapidly.

In the beginning, the percentage complete of work order is very low due to the slight of groups. After an enterprise technical report sharing, all the groups are encouraging to put much more efforts. In the end, the work orders are more reasonable to complete on time. This reflects the design of training courses correct the developing habits of students.

Fig. 5 shows the weekly work order of each project group. There is a general upward trend of weekly work order due to assignment check and periodical presentation. Some project groups contribute little work to this project at the beginning. We made some motivating measures to give some small gifts

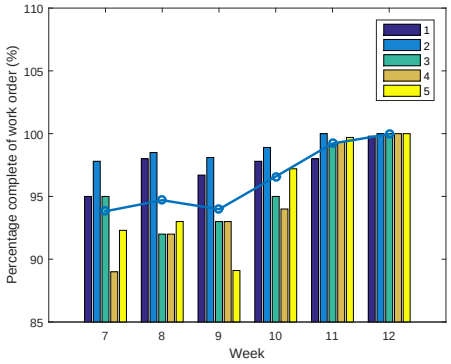

Figure 4: Percentage complete of work order of each project group.

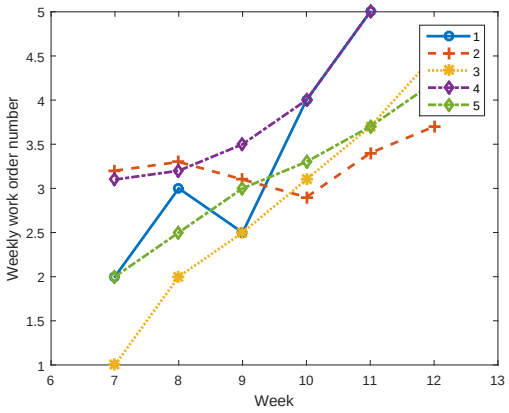

Figure 5: Weekly work order of each project group.

(e.g. earphone, mouse and USB flash disk) to the best group in the middle of a team project. This increases students engagement.

6.2 Time Estimation of Work Order

Time estimate cultivates the habit of finishing work as planned. Fig. 6 shows the percentage of work order that has normalized time estimate. As shown in Fig. 6, it is difficult to keep

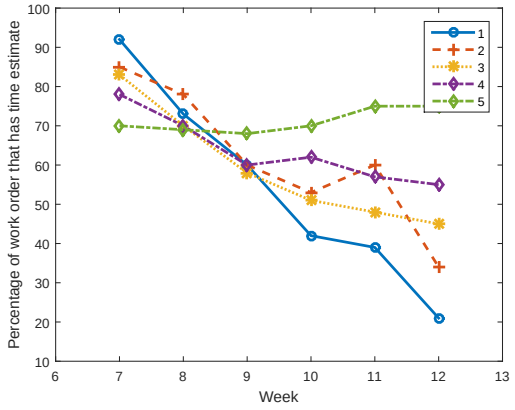

Figure 6: Percentage of work order that has normalized time estimate.

time estimate habit during the development process, especially in the cased of unplanned requirement.

6.3 Quality of Work Order

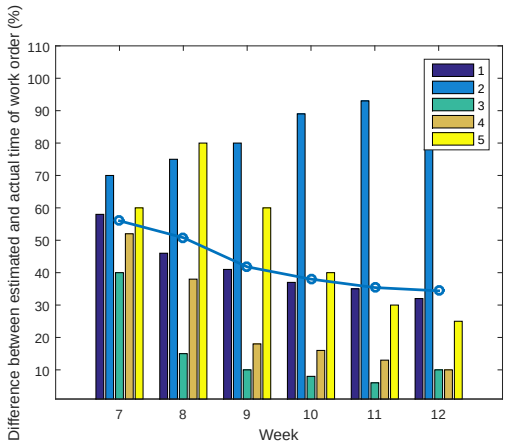

Figure 7: Difference between estimated and actual time of work order.

Fig. 7 shows the difference between estimated and actual time of work order of each group. The difference between estimated and actual time of work order of each group diminished over with time. This is due to the normalized schedule. Therefore, time estimate of work order increases the cognitive level of knowledge and the accuracy rates of development schedule.

We use SVN version control tool to manage the revisions. Fig. 8 shows the percentage of undesirable commit. The number of undesirable commit decreases with time due to normalized practice guidance. We show some normalized commit in the middle of engineering practice. As shown in Fig. 8, version control and collaborative tool is effective.

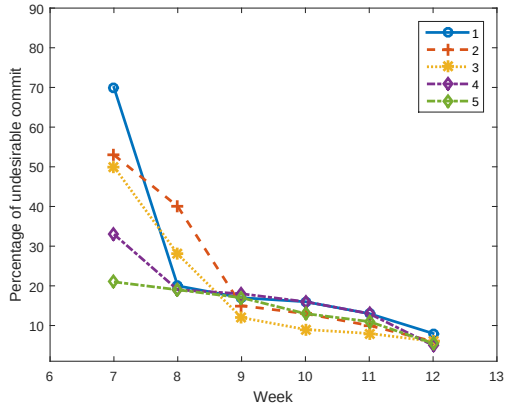

Figure 8: Percentage of undesirable commit.

6.4 Survey of Student Engagement

Table 2 shows the student engagement survey with some questions. Although this engineering training is difficult and tiring, most of students gained a lot at knowledge and experiences. Participation of enterprise engineer and normalized team project gives students a better training experience.

6.5 Feedback from Students

We now turn to feedbacks from students to evaluate the effect of our software engineering training. Table 3 shows some feedbacks from students. Most of students gain the harvest out of the books.

7 CONCLUSIONS

In the new era of Emerging Engineering Education (3E) in China, we propose outcome-based school-enterprise cooperative software engineering training to cultivate students to address complicated engineering issues. School-enterprise collaboration can increase student engagement of engineering practice effectively.

ACKNOWLEDGEMENT

This work was supported by the Industry-Academy Cooperative Education Project of Ministry of Education (201702185051 & 201701002017 & 201601018009), the National Natural Science Foundation of China (61772231), and the Shandong Provincial Natural Science Foundation (ZR2017MF025). I gratefully acknowledge financial support from Tencent.

REFERENCES

[1] Abdulrahman Alarifi, Mohammad Zarour, Noura Alomar, Ziyad Alshaikh, and Mansour Alsaleh. 2016. SECDEP: Software engineering curricula development and evaluation process using SWEBOK. *Information and Software Technology* 74 (2016), 114–126.

[2] Andre B Bondi. 2017. Experience with Performance Engineering Training in Academic and Industrial Environments. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*. ACM, 191–191.

Table 2: Agreement level of of several questions.

No.	Question	Score (out of 5 grades)
1	Master SVN version control tool.	4.78
2	Master unit testing method.	4.52
3	Requirement is as important as design.	4.89
4	Pair programming is more difficult than self development.	4.72
5	Like collaborative development more.	4.27
6	team members do a good job.	3.53
7	Code style is important.	4.92
8	Project review is important.	4.88
9	Like collaborative development more.	4.27
10	Intermittent presentation is important.	4.71
11	Learn more from intermittent presentation.	4.83
12	Know self insufficiency from intermittent presentation.	4.94
13	Suffer happiness despite hard work.	4.95
14	Learn new knowledge from practice.	4.89
15	Participation of enterprise engineer.	4.99

Table 3: Feedbacks from students.

No.	Feedback
1	know more about the software industry
2	improve the ability to communicate with others
3	learn with intention
4	find friends to defend practice
5	motivate interest, strengthen self learning, and have good mentality.
6	Clear division of work, collaborative code management.
7	Effort has harvested after several week's harvest effort.
8	Use SVN to manage versions and commits

[3] Alejandro Calderón, Mercedes Ruiz, and Rory V OConnor. 2017. ProDecAdmin: A Game Scenario Design Tool for Software Project Management Training. In *European Conference on Software Process Improvement*. Springer, 241–248.

[4] Pablo Delatorre and Alberto Salguero. 2016. Training to capture software requirements by role playing. In *Proceedings of the Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality*. ACM, 811–818.

[5] Vahid Garousi, Kai Petersen, and Baris Ozkan. 2016. Challenges and best practices in industry-academia collaborations in software engineering: A systematic literature review. *Information and Software Technology* 79 (2016), 106–127.

[6] Magne Jørgensen, Tore Dybå, Knut Liestøl, and Dag IK Sjøberg. 2016. Incorrect results in software engineering experiments: How to improve research practices. *Journal of Systems and Software* 116 (2016), 133–145.

[7] Vladimir G Khalin, Alexander V Yurkov, and Yury V Kosov. 2017. Challenges of the Digital Economy in the Context of Globalization: Training of PhDs in Software Engineering in Russia. In *International Conference on Digital Transformation and Global Society*. Springer, 120–129.

[8] Qiang Liu, Wentao Zhao, Dan Liu, and Jiao Ji. 2017. Evaluation of situational case-based teaching technique in engineering education. In *Proceedings of the ACM Turing 50th Celebration Conference-China*. ACM, 11.

[9] Eugene Yi Sheng Loh and Yuen Hang Ho. 2017. ICT Implementation and Its Effect in Engineering Education. *Journal of Emerging Global Technologies* 1, 2 (2017), 42–44.

[10] Kun Ma, Hao Teng, Lixin Du, and Kun Zhang. 2014. Project-driven learning-by-doing method for teaching software engineering using virtualization technology. *International Journal of Emerging Technologies in Learning (iJET)* 9, 9 (2014), 26–31.

[11] Kun Ma, Hao Teng, Lixin Du, and Kun Zhang. 2016. Exploring model-driven engineering method for teaching software engineering. *International Journal of Continuing Engineering Education and Life Long Learning* 26, 3 (2016), 294–308.

[12] Ke Mao, Licia Capra, Mark Harman, and Yue Jia. 2017. A survey of the use of crowdsourcing in software engineering. *Journal of Systems and Software* 126 (2017), 57–84.

[13] Bouchaîb Riyami, Khalifa Mansouri, and Franck Poirier. 2016. TOWARDS A HYBRID UNIVERSITY EDUCATION, INTEGRATION OF MOOCS IN INITIAL TRAINING PROGRAMS: A CASE OF A BIG PRIVATE EDUCATION STRUCTURE IN MOROCCO. In *INTED 2016*, Vol. 1. 6132–6141.

[14] Gordon W Skelton, Jacqueline Jackson, and F Chevonne Dancer. 2013. Teaching software engineering through the use of mobile application development. *Journal of Computing Sciences in Colleges* 28, 5 (2013), 39–44.

[15] Nikolai Tillmann, Jonathan De Halleux, Tao Xie, Sumit Gulwani, and Judith Bishop. 2013. Teaching and learning programming and software engineering via interactive gaming. In *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 1117–1126.

[16] Zhang Yi, Wen Jun-hao, Liu Ling, Xiong Qing-yu, and Gao Min. 2016. Study of Progressive Training Plan on Software Engineering Talents. In *Software Engineering Education Going Agile*. Springer, 15–20.