The Journal of Systems and Software 136 (2018) 139–152

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

A replicated experiment for evaluating the effectiveness of pairing practice in PSP education

Guoping Rong[a,b], He Zhang[a,*], Bohan Liu[a], Qi Shan[a], Dong Shao[a]

[a] Software Institute, Nanjing University, China
[b] The National Key Lab of Novel Technology, Nanjing University, China

ARTICLE INFO

Article history:
Received 31 March 2016
Revised 12 May 2017
Accepted 2 August 2017
Available online 26 August 2017

Keywords:
Personal software process
Software engineering education
Replication

ABSTRACT

Background: Handling large-sized classes is one of the major challenges in Personal Software Process (PSP) education in a tertiary education environment. We applied a *pairing* approach in PSP education and managed to mitigate the size challenge without sacrificing education effectiveness, which has been verified in an experiment in 2010 (*PSP2010*). However, there are several issues (e.g., mutual interference among student pairs, confusing evaluation comments, untraceable corrections, etc.) existing in this experiment, which may create mist towards proper understanding of the education approach.

Objective: In order to address the identified issues and better understand both pros and cons of the *pairing* approach, we replicated the experiment in 2014.

Method: With new lab arrangement and evaluation mechanism devised, the replication (*PSP2014*) involved 120 students after their first academic year, who were separated into two groups with 40 pairs of students in one group and 40 solo students in the other.

Results: Results of the replication include: 1) paired students conformed process discipline no worse (sometime better) than solo students; 2) paired students performed better than solo students in the final exam; 3) both groups spent comparable amount of time in preparing submissions; 4) both groups performed similar in size estimation and time estimation of the course assignments; 5) the quality of the programs developed by paired students is no less (sometime better) than solo students.

Conclusion: The replication together with the original study confirms that, as an education approach, the *pairing* practice could reduce the amount of submissions required in a PSP training without sacrificing (sometime improving) the education effectiveness.

1. Introduction

Personal Software Process (PSP), designed by Watts S. Humphrey, provides a framework of techniques to help engineers improve their performance (Humphrey, 2005). As stated by Humphrey, software engineers can enjoy PSP's benefits, such as fewer code defects, better estimation and planning, enhanced productivity by learning and using the process disciplines (Humphrey, 1996). In addition, the measurement and analysis tools included in the PSP framework can also help software engineers understand their own capabilities and hence improve personal performance. Hayes et al. conducted an empirical study of the impacts of PSP on individual software engineers during PSP training, and the results indicated that the improvements of the trainees can be expected in four dimensions: size estimation accuracy, effort estimation accuracy, product quality and process quality (Hayes and Over, 1997).

Due to the advantages mentioned above, it is believed that delivering PSP course to students in universities can help to train qualified software engineers who meet requirements from the industry better. However, teaching PSP in universities may easily become a big challenge due to typically large-sized classes and low students' participation. A typical PSP course is a full-time course which required all the students (trainees) to learn and complete it within 10 continuous days (normally in 2 weeks). The number of students in a regular class is no more than 25, and students are required to finish their assignments independently. Nevertheless, teaching PSP course in the tertiary education environment (especially in China) confronts totally different situation from the original course design. One of the major issues is the *class size*. Due to the rapid development of the Information Technology (IT) industry, many students poured into departments such as *Computer Science (CS), Software Engineering (SE)* and similar disciplines, rendering each class easily has hundreds of students in Chinese colleges.

* Corresponding author.
E-mail addresses: dr.hezhang@gmail.com, hezhang@nju.edu.cn (H. Zhang).

http://dx.doi.org/10.1016/j.jss.2017.08.011
0164-1212/

Along with the increasing number of students involved, the necessary real-time interaction between the lecturers and the students can be inevitably affected. Particularly, as a critical part of the process for learning PSP, timely and specific comments and feedback to the students' submissions are almost impossible. Students may lose the chance to correct their development behaviors deviating from a sound PSP process discipline.

In order to tackle this issue, we borrowed the concept of Pair Programming (PP) and adjusted it into a PSP course with a large-sized class in Nanjing University. One noteworthy adjustment is that we had fixed *driver/navigator* role within one assignment, which differs from regular PP. With this *pairing* practice, each pair of students could provide only one submission per pair instead of two individually; workload for both educators and students might be reduced by half so as to support a more productive assessment. However, since the philosophies, principles and values are different for PSP and PP (Boehm and Turner, 2003), it is not common to combine these two methods and hence there are very limited number of empirical studies on the effectiveness of incorporating PP into PSP education. This formed the immediate motivation of our research to explore the pros and cons of this integrated approach. We designed and conducted an experiment at Nanjing University in 2010 (*PSP2010*) to find the degree to which the typical issues in PSP courses with large scale could be solved by the *pairing* practice as well as the opportunities to improve this teaching approach. The preliminary results indicate that the performance of the paired students in assignments was significantly better than those of the solo students (Rong et al., 2011). However, the effectiveness of education (measured by scores in the final exam) has no significant difference between the two groups. More importantly, the feedback from both groups reflected some outstanding issues in *PSP2010*, including mutual interference among pairs, difficulty in understanding evaluation comments from different teaching assistants, and the inability of tracking corrections over submissions, and so on.

This motivated us to re-investigate the whole course process by replicating the experiment in 2014 (*PSP2014*). With the same process context defined by PSP, we followed the general experimental design of *PSP2010* in 2014 except several improvements (e.g., selection of subjects, lab environment and evaluation methods, etc.) as new treatments in *PSP2014*. The results of *PSP2014* not only reinforce the conclusions from *PSP2010* that applying *pairing* practice into PSP course will help to tackle issues such as large class size, but also more explicitly reveal several specific advantages of the teaching approach, e.g., better performance in final exams, better quality with the source code, etc.

The paper is structured as follows: Section 2 describes the research background and related work, in which a brief introduction of PSP and PP and the status of their adoption and research are included. In Section 3, we elaborate the original study, i.e. the research questions, design of the experiment and limitations of *PSP2010*. Section 4 describes the replicated experiment, i.e. *PSP2014*. The results and analyses are presented and compared in detail in Section 5. Based on the experimental results and experiences, some considerations and recommendations on how to adopt results in both the original study and the replication are discussed in Section 6. Besides, we also discuss some issues associated to the validity of our study in this section. Finally, the paper is concluded in Section 7 with suggestions for future work.

2. Background and related work

Both PSP and PP are widely included in curricula for CS/SE students in universities. This section introduces the background and the research related to PSP and PP.

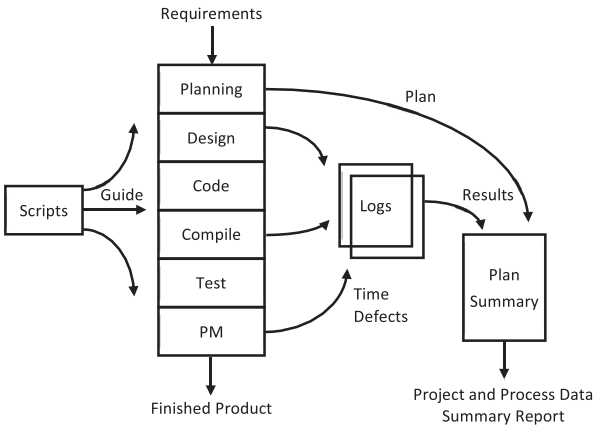

Fig. 1. PSP0 process flow (Humphrey, 2005).

2.1. Personal software process

Fig. 1 depicts the main framework of PSP (Humphrey, 2005). Six development phases are defined to develop software at the module (one-programmer) level. Scripts are used as guidelines for engineers to apply the PSP process consistently. Logs are used not only for measurement, but also serve as the footprint to determine whether the PSP process discipline is well followed by the programmers. Personalization is an essential feature of PSP; therefore, engineers who learn and practice PSP should record logs by themselves and use the information in the logs for personal process management and improvement (e.g., size and time estimation, defect analysis and prevention, etc.). According to various process elements contained, PSP could be introduced with upward-compatible levels, i.e., PSP0 (as presented in Fig. 1), PSP0.1, PSP1, PSP1.1, PSP2 and PSP2.1. The first four levels of PSP share the same development steps, however, two more development steps are introduced to PSP2 and PSP2.1, i.e., *Design Review* after *Design* and *Code Review* after *Code*.

Since its introduction, PSP has been used in higher education to provide students with self-improvement software process. It can serve as a part of other courses, or as a standalone course. In one experiment reported in Brown et al. (1997), PSP is used throughout the entire curricula. Alternatively, PSP can be contained in a series of courses related to SE. It is reported that some experiments integrated several typical practices of PSP into traditional programming courses (Lisack, 2000; Grove, 1998; Maletic et al., 2001). In all these cases, the PSP practices need to be tailored to meet various objectives of different courses as well as time constraints (Brown et al., 1997; Grove, 1998). The results from these attempts show that PSP brought some positive impacts to these courses, e.g., it can help solve the problems of poor estimation and planning (Brown et al., 1997; Maletic et al., 2001), shorten debugging time and improve quality (Grove, 1998), and increase productivity (Runeson, 2001).

However, since many practices and measures in PSP are interdependent, incomplete delivery of PSP course and partial practice of the PSP process are likely to inevitably impact the training effect. For instance, the attempts in education mentioned above (Brown et al., 1997; Lisack, 2000; Grove, 1998; Maletic et al., 2001) only provide training on basic PSP process elements such as process steps and corresponding metrics, which may contribute limited help to quality management at individual level. Besides, while delivering the PSP course in combination with other courses, students may pay more attention on other content, e.g., algorithms and language specifications, rather than the PSP process disci-

pline (Lisack, 2000). In other experiments, when PSP was optional for students, a considerable number of students may quit using PSP in the end (Borstler et al., 2002; Prechelt and Unger, 2000).

When PSP was made as a standalone course, the results of the PSP course varied across different studies. Typical positive effects such as improved effort estimation skills and product quality could be observed, as reported in Humphrey (1996); Runeson (2001); Abrahamsson and Kautz (2002). However, in Abrahamsson and Kautz (2006), the students in PSP course failed to achieve significant improvements on size and effort estimation, and only two third (20 out of 31) students passed the course. In all these reported studies with full PSP courses delivered, their class sizes remain small with normally only dozens of students enrolled.

2.2. Pair programming

Pair Programming (PP) involves two programmers sitting side by side, working collaboratively on the same design, algorithm, code or test with one shared computer. Within a pair one is the *driver*, who is responsible for designing, typing the code, and having control over the shared resource (e.g., computer, mouse, and keyboard); the other is the *navigator* or *observer*, who has responsibility for observing how the *driver* work in order to detect errors and offer ideas in solving a problem (Williams, 2001). The *driver* and *navigator* are encouraged to swap roles as necessary. According to a recent study of systematic reviews in software engineering (Zhang and Babar, 2013), PP is one of the most popular topics drawn attention among the SE community. Meanwhile, a systematic review conducted by Salleh et al. in 2011 also implies that PP is one of the main pedagogical tools to support cooperative learning in higher education (Salleh et al., 2011).

With a series of reported experiments, researchers observed many positive effects of PP on software development. On the social side, it helps form more active laboratory atmosphere than the traditional laboratory; under such friendly atmosphere, students will be more productive with less frustration (Nagappan et al., 2003; Williams and Upchurch, 2001) and enhance self-confidence and enjoyment (Williams and Upchurch, 2001; McDowell et al., 2003; Cockburn and Williams, 2000). On the technical side, PP can help improve individual's coding ability (McDowell et al., 2003; Sison, 2008), improve design quality (Williams and Upchurch, 2001), reduce the number of defects (McDowell et al., 2003; Cockburn and Williams, 2000), reduce risks (McDowell et al., 2003) and enhance the quality of the final program (Williams and Upchurch, 2001) by establishing collaboration among students. On the educational side, PP can also reduce number of cheating cases (Williams and Upchurch, 2001), help reduce the workload of educators, and ultimately improve the quality of teaching.

There are many factors that may affect the final teaching effects when PP is applied. Among which, the working environment may affect the results of PP significantly. Williams and Kessler reported that 96% programmers agreed that appropriate workspace layout was critical to their success (Williams and Kessler, 2000). It hereafter implies that the pairs should be able to comfortably sit next to each other as they work, and the sitting arrangement was critical to the success of PP practices Williams et al. (2008) and Chigona and Pollock (2008). In addition, proper feedback mechanism such as assignment evaluation can also impact the effectiveness of PP practices. Therefore, as Williams et al. suggested, the teaching staff should give guidance to encourage pairs to find answers on their own, rather than providing them with answers (Williams et al., 2008).

3. PSP2010: the original study

With the recognition of the importance of process related skills for software engineers, the PSP course is a compulsory course in Nanjing University. The usual size of an annual PSP class was around 260, which form a big challenge to deliver this course, particularly in submission evaluation. In order to tackle this challenge, in 2010 summer school, we arranged some students to complete the PSP assignments with pairs so as to reduce the total amount of the submissions to be evaluated. Through this *pairing* practice, we aimed to provide quality evaluation comments for the students, which is believed to be critical for students to correct process deviations.

However, the philosophies and benefits of PSP and PP are different. For example, PSP is a rather 'plan-driven' method, and as the name implies, calls for personalization. PP is a practice in Agile processes, which de-emphasize planning and call for self-organization at a team level. To what extent these two processes work well together and what is the impact on PSP education is therefore unclear. Therefore, we conducted the initial experiment by comparing the performance between paired students and solo students in terms of their submission quality and final exam scores. The results indicate that learning PSP in pairs could reduce the workload in submission evaluation without sacrificing the effect and quality of PSP education (Rong et al., 2011). This section describes *PSP2010* – the original experiment.

3.1. Research questions and hypotheses

The primary research objective of experiment *PSP2010* is to investigate the effectiveness and efficiency of PSP training by pairing students. We applied a Goal/Question/Metric (GQM) (Basili, 1992) approach to systematically identify research questions and corresponding metrics for this experiment (shown in Table 1).

Based on a GQM analysis, we may set up the following hypotheses for *PSP2010*:

Following PSP process discipline. Students learned PSP were supposed to follow the process discipline during the practicing phase. By applying evaluation checklists (cf. Section 3.2.4), PSP instructors and teaching assistants could identify process deviations in students' submissions by checking items in the evaluation checklists. In this sense, the number of process deviations (a.k.a. inconsistent items *DEV*) detected from a student's submission could reflect the degree that the student following PSP process discipline. Therefore, we could establish the hypothesis as:

$H_{0,\ DEV}$. *: There is no difference between the paired students and the solo students on DEV.*

Apparently, the fewer inconsistent items identified, the better. With *pairing* practice, two students may have chance to reduce the number of inconsistent items by 'double checking' their submissions. In this sense, we also established an alternative hypothesis:

$H_{1,\ DEV}$. *There are less inconsistent items in the submissions for the paired students than the solo students.*

Scores in the final exam. Exam scores (*SCORE*) are also important evidence to reflect how do students understand and master the PSP process. Since all paired students still need to take the final exam speratedly, and also, our original motivation to adopt the *pairing* practice is to reduce the workload in submission evaluation, we could accept comparable performance in final exam for both groups. Nevertheless, if paired students could perform better in the final exam, this may become one bonus to adopt the *pairing* practice in a PSP course. In this sense, we could establish the null hypothesis and the alternative hypothesis as the following:

Table 1
The GQM analysis in *PSP2010*.

Goal(s)	Question(s)	Metric(s)
To investigate the effectiveness and efficiency of PSP education by pairing students	Do paired students produce higher quality of the assignments than solo students by following PSP process discipline?	To measure the number of inconsistent items for students in both groups.
	Do paired students perform better than solo students in the final exam?	To measure the scores in the final exam of both paired students and solo students.

$H_{0,\ score}$. *: There is no difference between the paired students and the solo students in terms of SCORE.*

$H_{1,\ score}$. *: The paired students' SCORE is higher than the solo students'.*

3.2. Experiment description

3.2.1. Subjects and participants

The subjects and participants of *PSP2010* are:

Students. who enrolled in the PSP course when they finished their first academic year in university. In *PSP2010*, based on their registration numbers, the first 154 students were chosen to form the pairs. As a result, 77 student pairs were formed, and each student had a fixed partner during the entire course. The other 65 students were required to complete all the assignments individually. Each student had his/her assigned seat and PC in the laboratory.

PSP instructor. whose major responsibilities in the course were two-fold: to deliver lectures and to drive submission evaluation.

Teaching assistants (TAs). who were selected from those well-performed sophomore students in a PSP course in last year. Their major responsibility is to evaluate the students' submissions. With reference to a documented checklist (cf. Section 3.2.4), they need to find deviations from the PSP process discipline in the students' submissions, inform corresponding students the results and make sure that the students have corrected all the inconsistent issues. Therefore, each submission may be evaluated in several rounds by TAs until all the inconsistent items have been addressed by the owner–the student (or student pair) who submitted the assignment.

3.2.2. Treatments

In *PSP2010*, the *pairing* practice only happened in the lab sessions when students were required to finish assignments by following PSP. For instance, in one pair (say Student *A* and *B*), if the assignments with the odd numbers (1, 3, 5 and 7) were completed by *A* (the *driver*) with *B* as the *navigator*, then in the even numbers (2, 4, 6 and 8), the students swapped their roles, i.e., *B* acted as the *driver* and *A* worked as the *navigator*. The swapping must be validated by TAs, so as to ensure that all the students completed the equal numbers of assignments during the course.

To guide the *pairing* practice, we provided a brief instruction to paired students for the lab sessions. For a single pair (Student *A* and *B*), suppose that *A* acts as the *driver* and *B* as the *navigator*. The work in this case is exactly the same as that *A* completes the work individually, while *B*'s main task is to participate in the discussion and verify whether *A* carries out the work in accordance with the correct PSP process discipline (e.g., recording time logs, defect logs etc.) in all PSP phases. The detailed tasks of the roles in pairs in each specific phase are elaborated in Table 2. Note that, only the *navigator* was required to collect process data.

Apparently, this is not a typical PP as in some *agile* development processes. In our *pairing* practice, the *driver* and the *navigator* do not swap their roles within one assignment as frequently as needed. However, this arrangement is deliberately designed as a compromise to avoid the phenomenon that certain student pairs may heavily rely on one student to finish all the assignments. Nevertheless, the consequence of this arrangement may need further discussion (cf. Section 6.2).

3.2.3. Experimental design

In *PSP2010*, the 10-day PSP course and the experiment started in the summer school after the first academic year ended. The experiment process involves the following activities:

Lecture and practice. Both the schedule and content for the PSP course followed a standard PSP course designed by the Software Engineering Institute (SEI). We use classroom lectures combining laboratory practices to deliver a complete PSP training. A PSP instructor gave all the lectures and explained the requirements of each assignment to the whole class in one big classroom. After that, in the lab sessions students worked on their assignments in three large labs by following a certain PSP process (usually specified in the assignment requirement document). Each student had a dedicated seat and computer in the laboratories.

We used SEI's standard PSP materials[1] to deliver the course. The arrangements for the entire course are shown in Table 3. The serial numbers used in the table is the same as that in SEI's course materials. Note that there are no programming exercises on Day 5 and 10.

Assignment and evaluation. All assignments had to be completed after the daily lecture. As presented in Table 3, except the reports, students were required to write a program by following a specified level of PSP process (cf. Table A.13 in Appendix). Usually, students should finish and submit their assignments by the end of the day when they receive them.

Prior-evaluation training: Submission evaluation and feedback are extremely important for students to understand and master the PSP process. Therefore, the quality of the evaluation and feedback comments plays a vital role in a PSP course. The training before submission evaluation will help the TA team fully understand the assignment requirements and corresponding evaluation criteria. To guarantee the evaluation quality, we used a set of evaluation checklists (cf. Section 3.2.4) provided in the SEI's course materials. To train the TAs, a PSP instructor first explained each item in the checklist in detail and then demonstrated the use of the checklist with a representative submission (from former PSP courses) to the whole evaluation team. After that, the TA groups began to evaluate other submissions from former PSP courses, while the PSP instructor made necessary corrections.

Post-evaluation meeting: After each evaluation, the evaluation team also met for communicating the results and sharing their evaluation experiences. The typical topics at the meeting include (but not limited to):

- What are the typical issues in the students' submissions?
- Are there any cases difficult for the evaluation group to reach consensus?

[1] See http://sei.cmu.edu/tsp/tools/academic/index.cfm for details.

Table 2
Students' roles in pair.

Phase	Role	Task description
planning	A	to complete the size and time estimation of the program
	B	to participate in the discussion during the estimation steps.
detailed design	A	to complete the design.
	B	to observe the design process, make sure the design is in accordance with the predefined design template.
coding	A	to write the code and implement the design.
	B	to observe A's work, make sure the coding standards are followed. to review code for correctiveness.
unit testing	A	to complete the unit testing.
	B	to observe A's work and discuss the root cause of defects.
postmortem	A	to review process data together with B and find improvement opportunities.
	B	to review process data and discuss process improvement tips with A.

Table 3
The official PSP training program.

Day #	1	2	3	4	5	6	7	8	9	10
Lecture#	1	2	3	4	5	6	7	8	9	10
Assig. #	1	2	3	4	-	5	6	7	8	-
PSP ver.	0	0.1	1	1.1	-	2	2.1	2.1	2.1	-

- Are there any special circumstances that are worthy discussing?

Such discussions offered the evaluation team opportunities to further calibrate their evaluation criteria.

3.2.4. Experimental package

The course material provided by SEI forms the major portion of the experimental package. In addition, to collect evidence for further improvement on the course itself, we also provide each student with a form to collect feedback on both the lecture sessions and lab sessions of the PSP course.

Assignment requirements. According to the training plan, students need to finish eight programing assignments and two reports, which had to be completed after the daily lecture. As presented in Table 3, except Day 5 and Day 10, students were required to write a program (mostly mathematical calculation programs ranging between 50 ~ 300 LOC) by following a predefined PSP process (cf. Table A.13 in Appendix).

Recording logs. There are two types of recording logs (i.e. time recording log and defect recording log) defined in the PSP process. Both are included in the supporting tool provided by SEI course materials.[2]

Evaluation checklists. The evaluation checklists are also included in the course materials and are only used by the evaluators (i.e. the instructor and TAs). The items in the checklists mainly focus on the quality of the process data from various perspectives (e.g., the consistency and genuineness). Then based on the data, the evaluators could determine whether a student (or pair) has followed the process discipline in preparing submissions.

Feedback collection form. To collect more information so as to understand the quality of course delivery, we designed a questionnaire-based survey after the course. All the students were asked to answer several open questions regarding the course arrangements and facilities.

[2] http://sei.cmu.edu/tsp/tools/academic/index.cfm.

3.2.5. Data collection and analysis

Several metrics were defined using the GQM approach (as presented in Table 1), i.e. the number of inconsistent items and scores in the final exam respectively. All the data from paired students and solo students was collected for the final analysis.

1. *The number of inconsistent items (DEV):* The number of inconsistent items means the number of process deviations and issues detected by TAs according to the items on the corresponding checklist when evaluating a certain submission. However, we only collect the *DEV* in the evaluation to the first-round submission (not for corrections). The TA group collected this data daily.
2. *The scores in final exam (SCORE):* All students were tested with multiple-choice questions in the final exam. Here, *SCORE* is the mark (ranging from 0 to 100) each student achieved in the final exam. The TA group were responsible to collect this data after the PSP course.

Based on GQM analysis in Table 1, *DEV* can be used to test the hypotheses regarding following process discipline and *SCORE* can be used to test the hypotheses about students' performance in the final exam. However, to show the distribution of the data for both paired students and solo students, a boxplot was drawn to depict data *DEV* and *SCORE* for both groups. Besides, statistical tests are applied to test whether there is significant difference on *DEV* and *SCORE* between the two groups.

3.3. Results of PSP2010

3.3.1. Findings

Overall, the results of *PSP2010* indicated that the students working in pairs performed better than solo students on process discipline (Rong et al., 2011). However, there is no significant difference between two groups in terms of the scores in the final exam.

3.3.2. Limitations

The initial results from *PSP2010* imply several positive aspects to apply the *pairing* practice in a PSP course. However, there also exist some limitations of the experiment *PSP2010*, which may bring certain risks to reach a conclusive argument in favor of the *pairing* practice.

First, through an in-depth analysis of the checklist content, we found that it was not rare that several items come from an identical root cause. For instance, if a student failed to record the time log correctly, then he were going to get at least two inconsistent items against the evaluation checklist, i.e. *"time of the log is correct and reasonable"* and *"consistency on process data"*. Since an identical mistake might lead to inconsistency on both two items, it is more reasonable to combine these two inconsistencies as one. In this sense, the *DEV* data should be refined to reflect this phenomenon.

Next, some objectives of a PSP training were not well reflected in *PSP2010*, for example, the capability for size and time estima-

tion, the quality of the programs, etc. In this sense, the evaluation of the education approach is far from comprehensive.

Third, the feedback from both groups also implies some outstanding issues in the facilities of *PSP2010*, including mutual interference among pairs, difficulty in understanding the evaluation comments, and TA's insufficient tracking of students' corrections over submissions. For example, in *PSP2010*, students had each submission evaluated by different TAs. This arrangement, however, might lower TAs' evaluation consistency and disable them to track corrections over students' submissions, rendering students' inconsistent understanding of the PSP process discipline.

These issues may impact the performance of students in both lab sessions and the final exam, and as a consequence, the reliability of the analysis results in *PSP2010* may also be impacted to a certain degree.

4. The design of the replication

To further investigate and evaluate this education approach (i.e., the *pairing* practice), we conducted a replication study in 2014. The main motivations of the replication are two-fold:

1. To eliminate unintentional factors so as to better evaluate this education approach in a more rigorously controlled context.
2. To evaluate this education approach with more comprehensive perspectives for better understanding.

4.1. Commonalities with PSP2010

To address the above research motivations, we implemented an overall similar design of *PSP2010* in *PSP2014*. The common elements (with no change) shared by the both studies are as follows:

- *Same experimental package.* Both studies used the same experimental package as described in Section 3.2.4.
- *Same instructor.* The instructor in PSP2010 (also one of the author) participated in PSP2014.
- *Same timing.* Both studies happened in the summer school after students' first academic year within 10 continuous days.

4.2. Differences from PSP2010

As the replication study *PSP2014* also carried with the several changes (as presented in Table 4), compared to the original *PSP2010*. This section elaborates these changes as well as corresponding considerations in the replication.

4.2.1. Research questions and hypotheses

Following the GQM approach, we identified the research questions and metrics for *PSP2014* as shown in Table 5. Compared to the original, we added five more research questions to better evaluate the education approach.

Duration. If paired students spent significantly more time on assignments, it might discourage them applying the pairing approach in assignments. In contrast, if a significant amount of time could be saved through the pairing approach, it may also be an incentive for students to finish assignments in pairs. In this sense, we decided to examine the time spent on assignments for both groups. By measuring the duration of assignments in *PSP2014*, we established the null hypothesis and alternative hypothesis as follows:

$H_{0,\ \text{duration}}$. *: There is no significant difference between the paired students and the solo students on the time spent on assignments.*

$H_{1,\ \text{duration}}$. *: The paired students spent less time on assignments than the solo students..*

Size estimation. Improved estimation on both size and time is one of the claimed advantages of PSP education (Humphrey, 2005). For solo students, they can use their own historical process data in the former submissions for size and time estimation. On the contrary, the historical process data for the paired students came from two different sources (paired students should swap roles among different assignments), which may make it more difficult to be used in estimation. For example, if the abilities on programming and learning are significantly different between the two students in one pair, then the historical data works for one student may not be suitable for the partner in the same pair. However, a thorough discussion between the two students in one pair may also mitigate the difference and improve estimation accuracy (similar to a *Delphi* estimation technique). In this sense, we could establish the null hypothesis and alternative hypothesis as follows:

$H_{0,\ \text{size}}$. *: There is no significant difference between the paired students and the solo students on the accuracy of size estimation.*

$H_{1,\ \text{size}}$. *: The paired students can produce more accurate estimates on size than the solo students.*

Time estimation. For similar reason, we establish the hypotheses with respect to time estimation as follows:

$H_{0,\ \text{time}}$. *: There is no significant difference between the paired students and the solo students on the accuracy of time estimation.*

$H_{1,\ \text{time}}$. *: The paired students can produce more accurate estimates on time than the solo students.*

Program quality. Better program quality is also one of the claimed advantages for both PSP (Humphrey, 2005; 1996; Hayes and Over, 1997) and PP (Hannay et al., 2009). For paired students, the *navigator* could help the *driver* double check the programs, which may help the pair produce programs with higher quality. Since the program functionality is fixed for both groups, to facilitate the comparison, we use the number of defects (instead of defect density) identified in the unit test phase as the indicator of program quality. In this sense, we could establish the following hypotheses with respect to program quality:

$H_{0,\ \text{quality}}$. *: There is no significant difference between the paired students and the solo students on the quality of the programs.*

$H_{1,\ \text{quality}}$. *: The paired students developed programs with better quality than the solo students.*

4.2.2. Selection of students

In *PSP2010*, all the students were randomly selected to form the pairs. Although only those students (both paired and solo) who finished all the assignments were selected for final analysis, the large amount of submissions forced the TAs to conduct submission evaluation with separated mini-teams, rendering inconsistent evaluation criteria and difficulties to track students' corrections to the problematic submissions. In *PSP2014*, we reduced the number of students involved in the experiment and revised the evaluation process to address this issue.

Based on former experiences, on average, one TA could deal with around 30 submissions per day. Therefore, as a balance of the workload and the number of data points required to perform reliable statistical analysis, we set the number of data points as 40, which requires 40 students in the control group and 80 students (in pairs) in the experimental group. However, special arrangement on evaluation process is still required to handle the evaluation on 80 submissions each day (cf. Section 4.2.4). Therefore, by running a program, we randomly selected 120 students based on their registration numbers as the subjects in *PSP2014*.

Table 4
Changes to the original study.

Aspects	PSP2010	PSP2014	Motivation
Research questions & hypotheses	2 hypotheses regarding inconsistent items and scores in final exam	• 5 more research questions and hypotheses regarding evaluation on the education approach • one research question on the content of inconsistent items	• to evaluate the education approach more comprehensively • to identify improvement opportunities for future PSP courses
Selection of subjects	all students involved in the experiment	120 randomly selected students, who were separated into two groups: • 40 students in the control group • 80 students in the experimental group in pairs	to balance TAs' workload (for better evaluation) and the number of data points required for statistical analysis
Lab arrangement	one big lab was allocated for all the solo students and paired students	• More labs were allocated, to be specific, 2 labs were dedicated for the students involved in *PSP2014* • standardized hardware and software environment were set up before each lab session.	to control and reduce the unintended factors impacting the process data.
Evaluation process	assignments of solo student or student pair were evaluated by different TA teams.	• dedicated evaluation teams for all the submissions in the experiment • one submission was evaluated by different evaluators, each one focused on certain parts of items in the checklist.	• to allow consistent criteria and comments in the evaluation • to facilitate issues tracking among different assignments • to accelerate the evaluation speed
Data collection & analysis	2 types of data(i.e. 'the number of inconsistent items' and 'scores in the final exam')were collected and used for analysis	• we revised the definition of 'the number of inconsistent items' • we added 5 more types of data(i.e. 'duration of each assignment', 'relative error in size estimation', 'relative error in time estimation', 'defects in unit test', and 'category of inconsistent items')	• to evaluate the education approach more comprehensively • to identify improvement opportunities for future PSP courses

Table 5
The GQM analysis in *PSP2014*.

Goal(s)	Question(s)	Metric(s)	Comments
To investigate the effectiveness and efficiency of PSP education by pairing	Do paired students produce higher quality of assignments than solo students in terms of following PSP process discipline?	To measure the number of inconsistent items for both paired students and solo students in the predefined submission checklist in assignment evaluation.	Imported from *PSP2010*
	Do paired students perform better than solo students in the final exam?	To measure the scores in the final exam of both paired students and solo students.	Imported from *PSP2010*
	Do paired students spend more time than solo students in assignments?	To measure and compare the duration of each submission for both groups.	New in *PSP2014*
	Do paired students perform better on size estimation than solo students in assignments?	To measure and compare the relative errors on size estimation for both groups.	New in *PSP2014*
	Do paired students perform better on time estimation than solo students in assignments?	To measure and compare the relative errors on time estimation for both groups.	New in *PSP2014*
	Do paired students produce programs with better quality than solo students in assignments?	To measure and compare the number of defectsin each assignment for both groups.	New in *PSP2014*

4.2.3. Lab arrangement

First, we provided separated labs for two student groups, which allows generous space to ensure at least 2 m distance between solo students or pairs. Also the students were required to keep their discussions softly. With this arrangement, the mutual interference between two pairs could be minimized, which was confirmed in the students' feedback after the course. Second, in order to keep students' concentration on the assignments, we provided all students standardized hardware and software environment with the same configuration. With standardized facilities, the logistic effects to the recorded process data (e.g., issues with the PSP supporting tool, time spent in configuring IDE, defects caused by wrong version of JRE, etc.) could be well controlled.

4.2.4. Evaluation process

To ensure each submission evaluation was complete and consistent, we still used the standard evaluation checklist provided in the SEI course materials. However, compared to *PSP2010*, we made two important adjustments in *PSP2014*.

First, evaluators must provide *instructive evaluation comments* to students. The instructive evaluation comments need to address: 1) what are wrong in the submission? 2) what the correction should be? and 3) what are the possible reasons for this issue raising? The following is one typical example of the different types of evaluation comments in *PSP2010* and *PSP2014*, respectively.

Example of instructive comments

Item on SEI standard checklist: *Actual added on planning summary close to and no less than actual BA+PA on size estimating template.*
Typical evaluation comments in *PSP2010* : the student did not count A+M correctly
Typical evaluation comments in *PSP2014* :
Issue: did not count A+M correctly
Correct result: (A+M) in the project plan summary form should be close and no less than actual BA+PA on estimating template.
Reason: (A+M) in the estimating form only includes those code within methods in a Java class, while (A+M) in the project plan summary form also includes those code outside methods.

In *PSP2010*, TAs simply pointed out the mistakes, which may provide very limited information for students to understand the PSP process and make proper corrections. With the instructive evaluation in *PSP2014*, however, it can be more effective in helping

students understand PSP process discipline. This was confirmed by the students' feedback after *PSP2014* that the instructive evaluation was very helpful to understand and master the PSP principles and practices.

A fixed evaluation group (with two TAs and one PSP instructor) was assigned to evaluate all submissions of the students involved in the experiment. This adjustment allows consistent criteria and comments in the evaluation (from the same group). Besides, it also allows the evaluation group tracking the change history of each submission to ensure that all discovered issues had been understood and properly resolved. However, to accelerate the evaluation and supporting issue tracking, a checklist was split into three parts so that each evaluator normally only needed to focus on certain perspectives (i.e. certain items in the checklist) during evaluation.

4.2.5. Data collection and analysis

Based on the updated research questions and hypotheses, we first adjusted data *DEV* from the original study, and then added 4 new types of data in *PSP2014* to better characterize the students' performance in the PSP course:

1. Number of inconsistent items (*DEV*): The definition, collection and analysis of this data were the same as those in the original study, except that inconsistent items caused by an identical source should be combined and counted only once.
2. Assignment durations (*DUR*): *DUR* was collected by summarizing the actual time from the time recording log for all the PSP process steps. TAs calculated and collected this data.
3. Scores (*SCORE*): *SCORE* was collected based on students' scores in the final exam.
4. Relative error in size estimation (RE_S): According to the design, program size (in LOC) is not required for the first assignment. Therefore, both actual size and estimated size were collected from the rest assignments for all the submissions. RE_S could be calculated as Eq. (1) and compared between the solo students and the paired students.

$$RE_S = (A_s - E_s) \div A_s \tag{1}$$

where A_s denotes the actual program size (in LOC) and E_s means the estimated size.

5. Relative error in time estimation (RE_T): We could retrieve both estimated time and actual time of all submissions for the eight assignments. Similarly, RE_T could be calculated as Eq. (2).

$$RE_T = (A_t - E_t) \div A_t \tag{2}$$

where A_t denotes the actual time (in minute) spent on an assignment and E_t means the corresponding estimated time.

6. Defects fixed in unit test (*DEF*): The defect recording logs provide the data source for *DEF*. We define *DEF* as the number of defects removed in the Unit Test (UT) for each submission.

Furthermore, given the fixed program functionality and small size, we could group data points to better portray students' performance in our study. According to the course design, Assignments 1 through 4 require similar PSP processes. Therefore, we could combine the data points generated with these comparable PSP processes into one group. For the same reason, we also combined the data points generated in Assignments 5 through 8 into the other group. As the grouping results shown in Table 6, for Assignments $1\sim4$, we compared the data between G_{ss14} and G_{ps14} and G_{ss58} and G_{ps58} for assignments $5\sim8$. Note that size estimation is not required in the first assignment, therefore, the data points in G_{ss14} and G_{ps14} only contain three assignments (i.e. 2, 3 and 4) when the grouping mechanism was applied to compare RE_S.

Table 6
Experimental grouping.

		Including assignments	
		$1\sim4$	$5\sim8$
Different treatments	Solo students	G_{ss14}	G_{ss58}
	Paired students	G_{ps14}	G_{ps58}

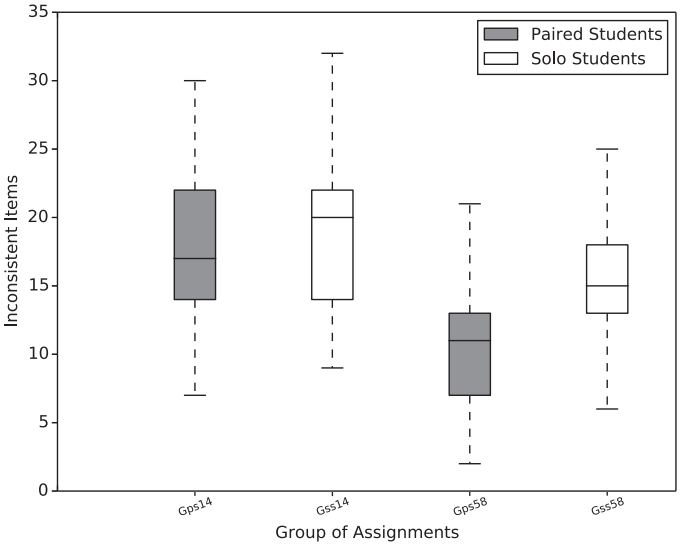

Fig. 2. Less inconsistent items occurred for paired students in *PSP2010*.

Table 7
Statistical test on *DEV*.

Group	PSP2010	PSP2014
–	*p*-value	*p*-value
G_{ss14} *VS.* G_{ps14}	**0.0018**	**0.0019**
G_{ss58} *VS.* G_{ps58}	**< 0.001**	0.713

5. Experimental results and analysis

Submissions with incomplete data occurred for students in both groups, which results 36 solo students and 33 pairs of students for the final analysis in *PSP2014*. We elaborate the analysis results and possible interpretations in this section.

5.1. Inconsistent items ($H_{0,\ DEV}$)

As depicted in Fig. 2, the *DEV*s (i.e. the number of inconsistent items) in *PSP2010* tended to vary between the paired students and solo students in the last four assignments. In contrast, in the first four assignments, the difference seems not apparent. However, the situation in *PSP2014* seems slightly different. As presented in Fig. 3, the paired students performed better than solo students regarding the number of inconsistent items in the first 4 assignments. Whereas, in the last 4 assignments, the difference seems not obvious.

We could also run a one-sided *Wilcoxon rank-sum test* to further test the difference. The results are presented in Table 7. With *p* values less than 0.05, we could reject $H_{0,\ DEV}$ and accept $H_{1,\ DEV}$ in *PSP2010* in favor of the paired students. Similarly, we can accept $H_{1,\ DEV}$ for the first four assignments in *PSP2014*, nevertheless, $H_{0,\ DEV}$ could not be rejected for the last four assignments in *PSP2014*.

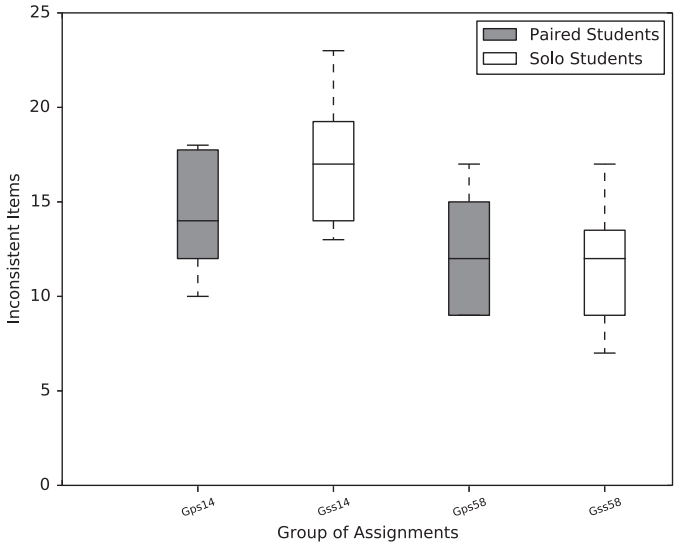

Fig. 3. Less inconsistent items occurred for paired students in the first 4 assignments in *PSP2014*, however, similar number of inconsistent items for both groups in the last 4 assignments.

Table 8
Statistical test on scores.

Study	*p*-value
PSP2010	0.333
PSP2014	**0.00015**

One possible reason for the above phenomenon is that in *PSP2010*, the content of inconsistent items were not merged according to their root causes, rendering potentially improperly counting of *DEV*s. With revised measurement on *DEV* in *PSP2014*, we might be able to present the status more accurately. In the first phase of a typical PSP course (i.e. the first four assignments) the students in both group were normally not familiar with the PSP process discipline and evaluation criteria as well. As the result, the paired students might possess the advantage of having each submission 'double checked'. However, along with the learning and practicing of PSP process, fewer inconsistent items occurred in the students' submissions for both groups. As the consequence, the *DEV* is then stabilized to a certain level for all the students.

Another notice is that the variation of *DEV* in *PSP2014* is smaller than that in *PSP2010*, which may to a certain degree better characterize the different performance between the control group and the experimental group. One reason contributing to this phenomenon might be the new approach to counting *DEV*. With only identical causes being counted, the process deviations (i.e. inconsistent items) tended to converge on several typical identical issues, which could lower the variation of *DEV*s for both groups.

5.2. *Scores* ($H_{0,\ score}$)

The scores in the final exam can be one of the measurable indicators to reflect the effects of PSP education in academic environment. In *PSP2010*, the paired students achieved similar scores as the solo students (67.17 vs. 68.65 on average) in the final exam. By applying a one-sided *Wilcoxon rank-sum test* on these two sample sets of scores, we got the results in Table 8, in which a p value of 0.333 suggested that we could not reject $H_{0,\ score}$ in *PSP2010*.

To better understand the status with respect to the scores in *PSP2014*, we used histogram portray the distribution of the scores for both groups. As depicted in Fig. 4 (the solo students) and Fig. 5 (the paired students), we could observe a slight difference between

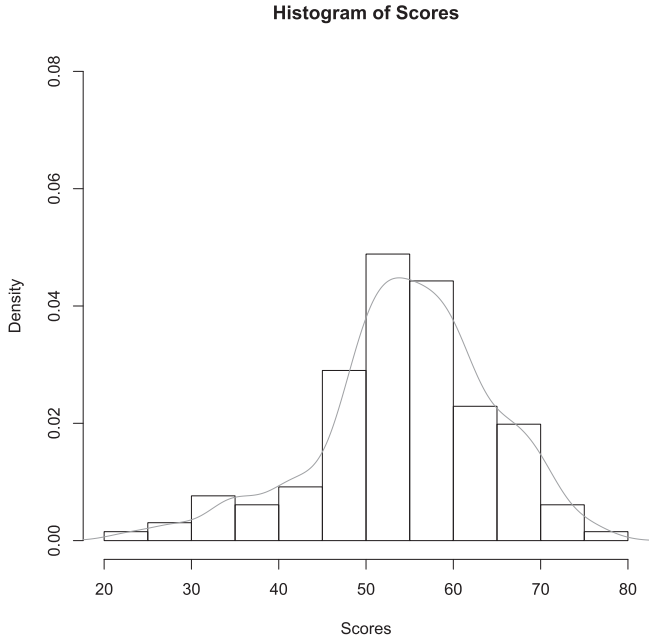

Fig. 4. Scores of solo students.

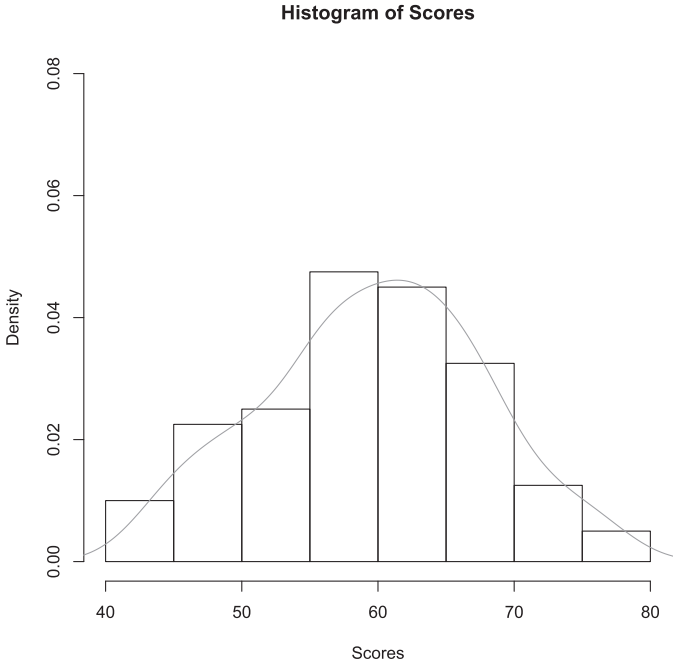

Fig. 5. Scores of paired students.

the two groups of students, and paired students seemed performing better than solo students. Similar, we applied the *Wilcoxon rank-sum test* on these two sample sets of independent observations. The results are presented in Table 8. With a p value of 0.00015 , we might reject $H_{0,\ score}$ and accept the alternative hypothesis $H_{1,\ score}$ at this time.

5.3. *Assignment duration* ($H_{0,\ duration}$)

DUR indicates the time spent for the whole assignment. As depicted in Fig. 6, the difference between the two groups for the first four assignments regarding duration is not quite clear. Meanwhile, in the last four assignments, the solo students seemed to spend slightly less time than the paired students. By applying a

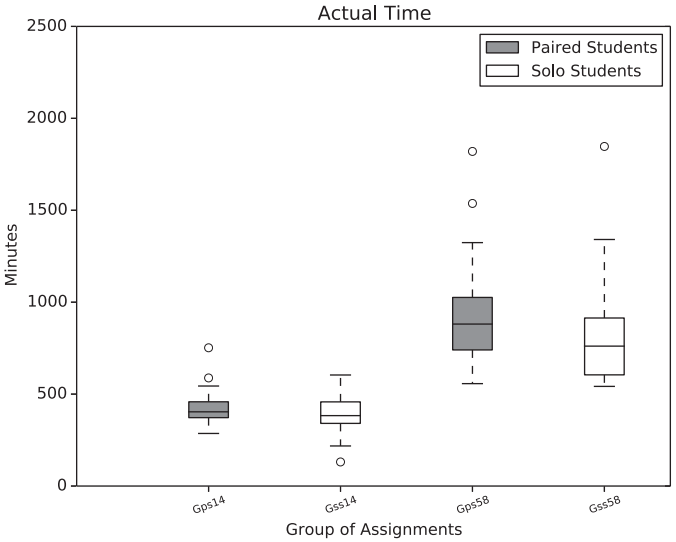

Fig. 6. Students spent similar amount of time in assignments.

Table 9
Statistical test on duration.

Group	*p*-value
G_{ss14} *VS.* G_{ps14}	0.841
G_{ss58} *VS.* G_{ps58}	0.957

Table 10
Statistical test on relative error of size estimation.

Group	*p*-value	
–	RE_S	$absRE_S$
G_{ss14} *VS.* G_{ps14}	0.89	0.698
G_{ss58} *VS.* G_{ps58}	0.435	0.903

one-sided *Wilcoxon rank-sum test* on *DUR*s for the two sets of observations on the duration of assignments, we have the results in Table 9. With all the *p* values are larger than 0.05, we can not reject $H_{0,\ duration}$. That is, we can not observe statistical difference between two groups with respect to the time spent in assignments.

One possible reason of this phenomenon is that for relative simple and small programming tasks, advantages such as higher productivity (i.e. smaller *DUR*) might not be able to be observed (Hannay et al., 2009). On the other hand, new learners also need time to get used to the PSP process, especially in the first five assignments with new PSP elements introduced in each assignment (cf. Table 3). When working solely, students may require extra time to get used to the new PSP process elements as well as to understand assignment requirements in detail. Meanwhile, when working in pairs, the *navigator* may help and support the *driver* to adopt the PSP process elements, which may make the paired students working faster than the solo students. In this sense, with confounded influences from both side, it is difficult to differentiate the two groups in terms of *DUR*.

5.4. Size estimation ($H_{0,\ size}$)

As presented in Fig. 7, the relative error on size estimation for both groups tends to be similar regarding the estimating trends (i.e., overestimating or underestimating). The results of the statistical test (shown in Table 10) also confirm the finding with all the *p* values higher than 0.05. In this sense, we can reject $H_{1,\ size}$ but cannot reject $H_{0,\ size}$.

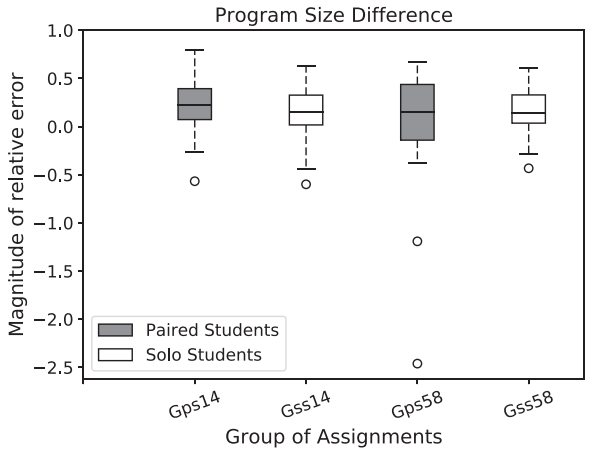

Fig. 7. Size estimation for both groups is similar in terms of relative error.

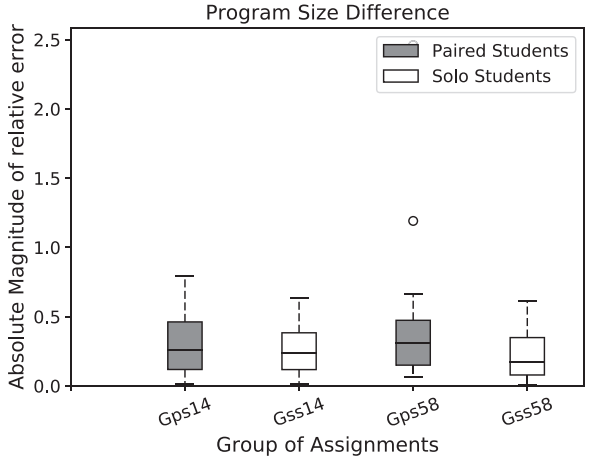

Fig. 8. Size estimation for both groups is similar in terms of absolute relative error.

The absolute relative error ($absRE_S$, calculated with Eq. (3)) may also provide a way to portray the estimation accuracy.

$$absRE_S = |A_s - E_s| \div A_s \tag{3}$$

where A_s **denotes the actual size (in LOC) of the programs and** E_s *means the estimated size.*

Fig. 8 depicts the distribution of the $absRE_S$s for all the assignments, which also looks similar for both groups. The results of the statistical test (as shown in Table 10) confirm this observation with all the *p* values higher than 0.05. Therefore, we could not reject $H_{0,\ size}$.

5.5. Time estimation ($H_{0,\ time}$)

Similarly, we could test $H_{0,\ time}$ as the same way to test $H_{0,\ size}$. In this sense, we defined the absolute relative error ($absRE_T$) as Eq. (4).

$$absRE_t = |A_t - E_t| \div A_t \tag{4}$$

where A_t *denotes the actual time (in minute) for each assignment and* E_t *means the corresponding estimated time.*

As presented in Figs. 9 and 10, the distribution of the relative error on time estimation (both RE_T and $absRE_T$) also tended to be alike.

Table 11 lists the results of a one-sided Wilcox test on RE_T and $absRE_T$ for both groups. All the *p* values were higher than 0.05, which suggests that $H_{0,\ time}$ could not be rejected in these occasions.

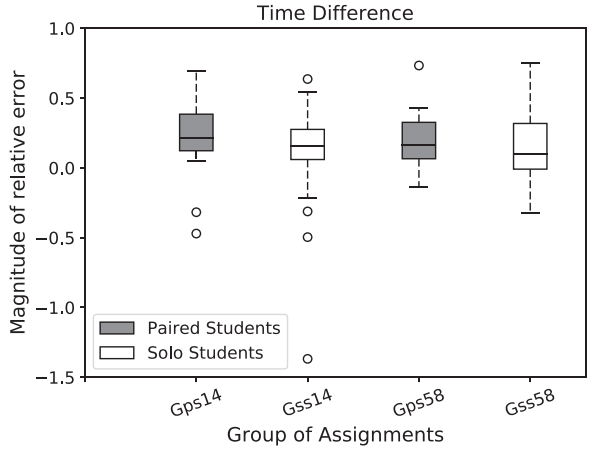

Fig. 9. Time estimation for both groups is similar in terms of relative error.

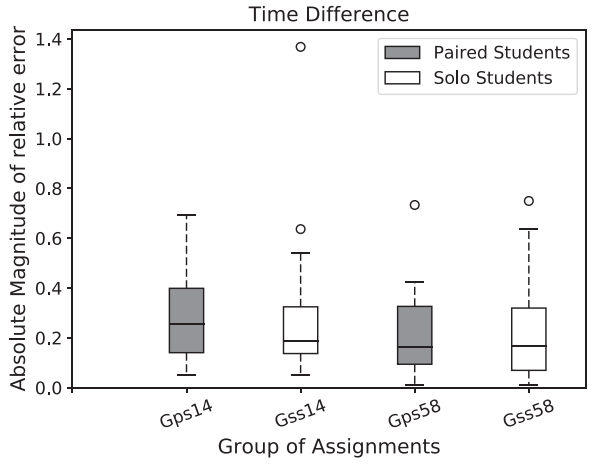

Fig. 10. Time estimation for both groups is similar in terms of absolute relative error.

Table 11
Statistical test on relative error of time estimation.

Group	*p*-value	
–	RE_T	$absRE_T$
G_{ss14} *VS.* G_{ps14}	0.928	0.824
G_{ss58} *VS.* G_{ps58}	0.827	0.576

Table 12
Statistical test on the number of defects in UT for students.

Group	*p*-value
G_{ss14} *VS.* G_{ps14}	0.419
G_{ss58} *VS.* G_{ps58}	0.141

5.6. Program quality ($H_{0,\ \text{quality}}$)

As presented in Fig. 11, in the first four assignments, students in both groups performed similarly in terms of the number of defects removed in the Unit Test (UT) phase. Meanwhile, there seems to be a slight difference between two groups in the last four assignments. However, a one-sided *Wilcoxon rank-sum test* on the difference can not support this intuitive observation. As presented in Table 12, with both p values higher than 0.05, we may not be able to reject $H_{0,\ quality}$.

However, with relatively simple programing tasks for all the 8 assignments, it was not expected a considerable number of defects

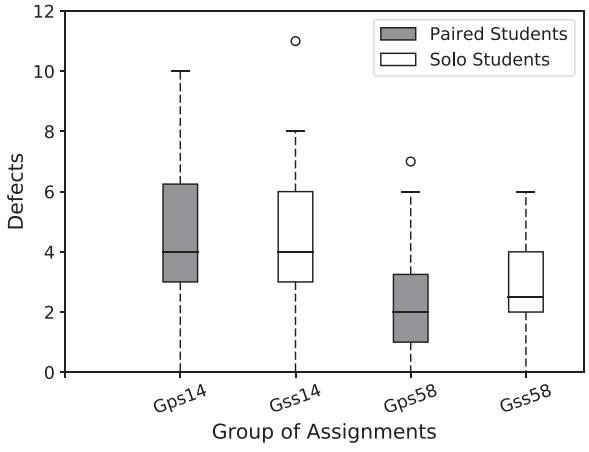

Fig. 11. Similar number of UT defects occurred for both groups.

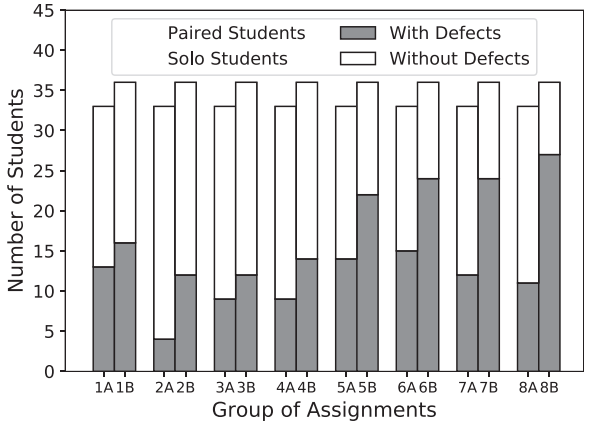

Fig. 12. Bigger proportion of students in pairs encountered no defects in UT.

being detected in UT for most students. As a matter of fact, very few students had more than one defects removed in UT in their submissions. In this sense, we provide another angle to characterize the students' performance in terms of program quality. As shown in Fig. 12, we present the proportion of students who were with/without defects in UT. For each pair of bars (assignment), the left presents the number of students in the experimental group (33 in group A), of which the (blank) top denotes the number of students without defects in UT; similarly, the number of students without defects identified in UT in the control group (36 in group B) is shown by the top of the right bar in each pair. Intuitively, paired students seemed to perform better than the solo students.

One possible reason is that when working in a pair, the *navigator* usually acted as a peer reviewer to help detect possible defects in the source code. However, this effect might not be obvious and significant for very simple programming tasks carried out by PSP new learners, which deserves further investigation.

6. Discussion

To tackle the challenge of big class size in the PSP course in tertiary education environment, we adopted a *pairing* practice in PSP education. We designed and executed an experiment to investigate the impacts of this new approach on PSP education in 2010 and replicated the experiment in 2014. The results in both experiments showed and confirmed the effectiveness of the *pairing* practice in dealing with the challenges without sacrificing the education quality. This section discusses the considerations related to this education approach as well as the implications of this replicated study.

6.1. Combined findings from PSP2010 and PSP2014

Based on the data and corresponding analysis *PSP2010* and *PSP2014*, we may summarize the findings as follows:

The pairing practice reduced the workload for submission evaluation. This is the direct and also obvious result of the *pairing* practice, which may allow instructors to handle relatively large class size. As a direct advantage of the fact, the effective feedback mechanism (a.k.a. the evaluation and issue tracking) could still be feasible.

The paring practice could help students understand and follow PSP process disciplines better. This fact is more obvious with new PSP learners (as shown in the early stage of the assignments in both *PSP2010* and *PSP2014*). However, when students get more and more familiar with the PSP process disciplines (as indicated by relatively small *DEV* in the last four assignments), the advantage of the *pairing* practice regarding process conformity is not as obvious as the early stage.

The pairing practice could not reduce the time required for students to prepare submissions. For education purpose, this fact should arouse certain consideration due to the possible de-motivation to apply the *pairing* practice in PSP course. Meanwhile, this fact also to a certain degree confirms one of the commonly accepted conclusions on PP that this practice could not improve productivity on simple programing tasks (Hannay et al., 2009).

Estimating was difficult for students, no matter working in pairs or solely.. The students in both groups performed similarly regarding size estimating and time estimating. Based on the analysis in Sections 5.4 and 5.5, the improvements from the start till the end of all the lab sessions were not obvious.

The pairing practice could to a certain degree improve the program quality. Although the evidence confirming this fact is not so strong and solid in *PSP2014*, we deem it important to further study this finding, given the importance of quality in software projects.

6.2. Threats to validity

This section discusses the various threats to the validity of this replicated experiment.

6.2.1. Construct validity

Construct validity reflects the degree to which the measures in the study accurately represent the research intention. The following issues associated with construct validity have been identified:

Metrics to assess the quality of PSP education. The *pairing* practice could reduce the amount of workload for instructors, TAs and students by reducing the number of submissions. However, the six types of metrics in this study may only reflect certain perspectives to assess the quality of PSP education. For example, even the paired students performed well in the PSP course, to what degree they could still present good performance in a Team Software Process (TSP) team is not clear. In this sense, the motivation of our study may not be fully satisfied.

Relative error of estimates. We used the relative error as the indicator to compare the students' performance with respect to estimation. However, it might be insufficient to reflect the education objective, i.e. to help students improve the capability for better estimation. Software estimation is a process with many factors that might impact the estimating results. In this sense, even if the paired students estimated better than the solo students, it might be of risk to attribute the sole credit to PSP education. Hence, further investigation is expected.

Scores as the indicator to education effectiveness. In the final exam, all students were tested with multiple-choice questions only. However, multiple-choice questions may have limited power to reveal education effects comprehensively. According to Bloom's taxonomy of educational objectives (Bloom et al., 1956; LeBlanc et al., 2006), multiple-choice questions are more suitable to test low level skills in the cognitive domain, e.g., knowledge and concepts of PSP. We will seek more effective methods in evaluating education effects but keep potential bias well-controlled.

Size of the assignments in experimental package. All the assignments in the original study and the replication are relatively small in size due to the limited functionality (simple programming tasks). This fact may impact the experiments from several aspects, such as 1) the necessity to swap roles during the development, which was not allowed in our experiments, 2) the more possibilities to make large relative error of the estimates, and 3) the possible defects in source code and the difficulty to remove them as well.

Severeness of the inconsistent items. We did not consider the severeness of the inconsistent items when collecting and analyzing the *DEV*. This fact may impact the conclusions derived from our experiments to a certain degree. Nevertheless, unlike *PSP2010*, we combined and summarized the inconsistent items according to the identical source in *PSP2014*, by which the impacts from various severeness of the inconsistent items may be well mitigated.

6.2.2. Internal validity

Internal validity reflects the degree to which conclusions can be drawn about the causal effect of the independent variable on the dependent variables. The following issues associated with internal validity have been identified:

Violation to the blinding criteria. The submissions for assignments contain students' names, which did not support the blinding criteria suggested in typical experiments. However, all TAs in *PSP2014* were asked to strictly follow a standard checklist and marking guide that were provided by the instructor in finding the inconsistent items. The use of checklist and guide mitigated the potential evaluation bias due to the exposure of student's name.

Subjective bias. In order to test the hypotheses in both experiments, we collected the data in terms of the metrics presented in Tables 1 and 5. Most of the measures were based on the evaluation of students' assignments, where TAsâ;; subjective bias might be introduced.

Self-recorded process data. One more threat to internal validity may come from the time log recording. The data of assignment duration were calculated from the time-logs that were recorded by the students themselves. However, to minimize the mistakes in recording, in the early lab sessions, the PSP instructor worked on-site and provided guidance to help students establish good habits to track and record their time. Besides, there is also no indication that such recording errors were more frequent or significant in one group than in the other.

Different survival rates. In order to obtain complete data for the final analysis, we removed submissions with incomplete data. As the result, the data from 4 solo students and 7 pairs of students was discarded. In some way, the different survival rates between two groups may also reflect education effectiveness. And the results in *PSP2014* was in favor of the solo students. Nevertheless, there usually were complex reasons why students submitted incomplete data in PSP education. We did not investigate the different survival rates at this stage, which raises the necessity to conduct more replications and explore this phenomenon.

6.2.3. External validity

External validity is the degree to which the results of the research can be generalized to the population under study and other research settings. The following issues associated with external validity have been identified:

The period of course. The PSP course described in this paper takes ten continuous days. In academic environment, however, a typical course usually spans over one semester. The biggest challenge in stretching this course design to fit regular semester setting is how to schedule the sessions in a weekly manner. The current course consists of a number of three hour (lecture or lab) sessions, which can be scattered into each week of the semester. The lecture and lab sessions can be interchangeable. Considering the mean time of each assignment is around 150 minutes, lab session can be provided with multiple batches with a smaller number of students in each. This arrangement may allow more space and further reduce mutual interference. Hence, the full course can be completed in ten weeks, which can fit into one semester in university.

The pairing practice and PP. To certain degree, the *pairing* practice in our studies is not a typical PP practice. For example, in typical PP practice, developers were encouraged to swap roles (between *driver* and *navigator*) whenever necessary, yet in our experiments, the paired students could not swap roles in one assignment. In this sense, the results of this study may need further investigation before applying in typical PP.

7. Conclusions

Personal Software Process (PSP), as a widely accepted process to train software practitioners with management skills at individual level, can equip students with tactical development skills that are needed in industry. However, the success of PSP education in academic environments needs carefully-crafted approaches to tackling the challenge such as large-sized class. This paper reports an experiment and its replication to investigate an education approach by adopting typical practices of Pair-Programming (PP) into PSP course as an attempt to solve the oversize challenge. This approach aims to significantly relieve the heavy workload on submission evaluation and to improve the overall education effects as well.

Based on the results generated from this research, the main contributions of this study are three-fold. First, the results confirm the positive impacts of applying the *pairing* practice in a PSP course by testing hypotheses from various perspectives, and indicate the value of *pairing* practice to PSP education. Second, as an experimental replication on PP, this study also supports some advantages of PP claimed in many other studies with the comparable results between the experiment and the replication. Third, the combination of the PP and PSP in this study could also to a certain degree imply the possibility to combine processes/practices with different philosophies, principles, and values, which may better support modern software development.

Our research and experience at this stage also raise a few interesting topics for future research, such as:

- To comprehensively validate the effect of the PSP education approach proposed in this work. The measurement and analysis conducted in *PSP2010* and *PSP2014* focused on the learning stage of PSP only. However, the objective of PSP education is to prepare team members for TSP teams. It is necessary to further validate the education effect within TSP teams, especially for those students who completed the PSP course in pairs.
- To explore and improve this education approach in industrial environments. This approach has shown its value in academic education environment. In industrial settings, PSP training may also encounter challenges, such as the large number of software practitioners in some countries like China. However, it may be more difficult to find a sufficient number of TAs required in industry training than in university.
- To apply regular PP into PSP education. The *pairing* practice in this study is not a typical PP practice. While programmer pairs in typical PP may possess different dynamics, it is interesting to carry out empirical studies to investigate a typical PP practice in the training and practical adoption of PSP.

Appendix A. Programing requirements for 8 assignments

Table A.13
Programming requirements.

Assignment #	Requirement specification	Process
1	Using PSP0, write a program to calculate the mean and standard deviation of a set of n real numbers.	PSP0
2	Using PSP0.1, write a program to count (in LOC) the • total program size • total size of each of the program's parts (classes, functions, or procedures) • the number of items (or methods) in each part	PSP0.1
3	Using PSP1, write a program to • calculate the linear regression parameters β_0 and β_1 and correlation coefficients $r_{x,y}$ and r^2 for a set of n pairs of data, • given an estimate, x_k calculate an improved prediction, y_k where $y_k = \beta_0 + \beta_1 \times x_k$	PSP1
4	Using PSP1.1, write a program to calculate relative size ranges for very small, small, medium, large, and very large ranges using standard deviation.	PSP1.1
5	Using PSP2, write a program to numerically integrate a function using Simpson's rule. Use the t distribution as the function.	PSP2
6	Using PSP2.1, write a program to find the value of x for which integrating the t function from 0 to x gives a result of p.	PSP2.1
7	Using PSP2.1, write a program to • calculate the correlation between two sets of numbers x and y • calculate the significance s of that correlation • calculate the linear regression parameters β_0 and β_1 for a set of n pairs of data, • given an estimate,x_k calculate an improved prediction, y_k where $y_k = \beta_0 + \beta_1 \times x_k$ • calculate the 70% prediction interval for that estimate	PSP2.1
8	Using PSP2.1, write a program to calculate the three-variable multiple-regression estimating parameters (β_0, β_1, β_2, β_3). Make an estimate from user-supplied inputs, and determine the 70% prediction intervals for the estimate.	PSP2.1

References

Abrahamsson, P., Kautz, K., 2002. The personal software process: experiences from denmark. In: The 28th Euromicro Conference, Proceedings.. IEEE, pp. 367–374.

Abrahamsson, P., Kautz, K., 2006. Personal Software Process: Classroom Experiences from Finland. Springer.

Basili, V. R., 1992. Software modeling and measurement: the Goal/Question/Metric paradigm.

Bloom, B.S., of College, C., Examiners, U., 1956. Taxonomy of Educational Objectives, 1. David McKay New York.

Boehm, B., Turner, R., 2003. Balancing Agility and Discipline: A Guide for the Perplexed. Addison-Wesley Professional.

Borstler, J., Carrington, D., Hislop, G.W., Lisack, S., Olson, K., Williams, L., 2002. Teaching PSP: challenges and lessons learned. IEEE Softw. 19 (5), 42–48.

Brown, A., Oudshoorn, M., Maciunas, K., 1997. The personal software process in undergraduate software engineering education. In: Proceedings of International Symposium on Software Engineering Education in Universities, Rovaniemi, Finland. Citeseer, pp. 52–59.

Chigona, W., Pollock, M., 2008. Pair programming for information systems students new to programming: Students experiences and teachers challenges. In: Portland International Conference on Management of Engineering&Technology, PICMET2008.. IEEE, pp. 1587–1594.

Cockburn, A., Williams, L., 2000. The costs and benefits of pair programming. Extreme Program. Examined 223–247.

Grove, R.F., 1998. Using the personal software process to motivate good programming practices. In: ACM SIGCSE Bulletin, 30. ACM, pp. 98–101.

Hannay, J.E., Dybå, T., Arisholm, E., Sjøberg, D.I., 2009. The effectiveness of pair programming: A meta-analysis. Inf. Softw. Technol. 51 (7), 1110–1122.

Hayes, W., Over, J.W., 1997. The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers.. Technical Report. DTIC Document.

Humphrey, W.S., 1996. Using a defined and measured personal software process. IEEE Softw. (3) 77–88.

Humphrey, W.S., 2005. PSP: A Self-Improvement Process for Software Engineers. Addison-Wesley Professional.

LeBlanc, R.J., Sobel, A., Diaz-Herrera, J.L., Hilburn, T.B., et al., 2006. Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering. IEEE Computer Society.

Lisack, S.K., 2000. The personal software process in the classroom: student reactions (an experience report). In: The 13th Conference on Software Engineering Education & Training. IEEE, pp. 169–175.

Maletic, J.I., Marcus, A., Howald, A., 2001. Incorporating PSP into a traditional software engineering course: an experience report. In: CSEE & T. IEEE, p. 89.

McDowell, C., Werner, L., Bullock, H.E., Fernald, J., 2003. The impact of pair programming on student performance, perception and persistence. In: Proceedings of the 25th International Conference on Software Engineering. IEEE Computer Society, pp. 602–607.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., Balik, S., 2003. Improving the CS1 experience with pair programming. In: ACM SIGCSE Bulletin, 35. ACM, pp. 359–362.

Prechelt, L., Unger, B., 2000. An experiment measuring the effects of personal software process (PSP) training.

Rong, G., Zhang, H., Chen, Z., Shao, D., 2011. Delivering PSP course in tertiary education environment: Challenges and solution. In: 24th IEEE-CS Conference on Software Engineering Education and Training (CSEE&T). IEEE, pp. 284–293.

Runeson, P., 2001. Experiences from teaching PSP for freshmen. In: The 14th Conference on Software Engineering Education and Training, 2001. IEEE, pp. 98–107.

Salleh, N., Mendes, E., Grundy, J., 2011. Empirical studies of pair programming for CS/SE teaching in higher education: a systematic literature review. Softw. Eng. IEEE Trans. 37 (4), 509–525.

Sison, R., 2008. Investigating pair programming in a software engineering course in an asian setting. In: Proceedings of the 15th Asia-Pacific Conference on Software Engineering, 2008. APSEC'08.. IEEE, pp. 325–331.

Williams, L., 2001. Integrating pair programming into a software development process. In: Proceedings of the 14th Conference on Software Engineering Education and Training. IEEE Computer Society, p. 27.

Williams, L., McCrickard, D.S., Layman, L., Hussein, K., 2008. Eleven guidelines for implementing pair programming in the classroom. In: AGILE'08. IEEE, pp. 445–452.

Williams, L., Upchurch, R.L., 2001. In support of student pair-programming. In: ACM SIGCSE Bulletin, 33. ACM, pp. 327–331.

Williams, L.A., Kessler, R.R., 2000. All I really need to know about pair programming I learned in kindergarten. Commun. ACM 43 (5), 108–114.

Zhang, H., Babar, M.A., 2013. Systematic reviews in software engineering: an empirical investigation. Inf. Softw. Technol. 55 (7), 1341–1354.