Assessing PSP effect in training disciplined software development: A Plan–Track–Review model

Wen-Hsiang Shen a, Nien-Lin Hsueh a, Wei-Mann Lee b,*

a Department of Information Engineering and Computer Science, Feng Chia University, 100 Wenhwa Road, Taichung 40724, Taiwan, ROC
b Department of Information Technology, The Overseas Chinese University, 100 Chiaokwong Road, Taichung 40721, Taiwan, ROC

A R T I C L E   I N F O

Article history:
Received 17 November 2009
Received in revised form 2 August 2010
Accepted 28 September 2010
Available online xxxx

Keywords:
Personal software process (PSP)
Software process improvement (SPI)
Effect assessment
Plan–Track–Review

A B S T R A C T

Context: In training disciplined software development, the PSP is said to result in such effect as increased estimation accuracy, better software quality, earlier defect detection, and improved productivity. But a systematic mechanism that can be easily adopted to assess and interpret PSP effect is scarce within the existing literature.

Objective: The purpose of this study is to explore the possibility of devising a feasible assessment model that ties up critical software engineering values with the pertinent PSP metrics.

Method: A systematic review of the literature was conducted to establish such an assessment model (we called a Plan–Track–Review model). Both mean and median approaches along with a set of simplified procedures were used to assess the commonly accepted PSP training effects. A set of statistical analyses further followed to increase understanding of the relationships among the PSP metrics and to help interpret the application results.

Results: Based on the results of this study, PSP training effect on the controllability, manageability, and reliability of a software engineer is quite positive and largely consistent with the literature. However, its effect on one's predictability on project in general (and on project size in particular) is not implied as said in the literature. As for one's overall project efficiency, our results show a moderate improvement. Our initial finding also suggests that a prior stage PSP effect could have an impact on later stage training outcomes.

Conclusion: It is concluded that this Plan–Track–Review model with the associated framework can be used to assess PSP effect regarding a disciplined software development. The generated summary report serves to provide useful feedback for both PSP instructors and students based on internal as well as external standards.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Ever since its introduction by Humphrey at the Software Engineering Institute (SEI) in 1995, the personal software process (PSP) has been adopted and proven effective in software engineering training in both academic and industrial environments [1]. Its effects, in terms of software process improvement (SPI), include better project estimation accuracy, fewer defects, defects caught earlier, and improved productivity [2–4]. When PSP is used on campus, it can not only motivate students, benefit instructors, but also be adopted in both undergraduate and graduate software engineering courses [4–7]. Some even argue that with a modest disciplined SPI, a substantial effect can be generated from PSP training [8–10].

Regardless of these positive effects, however, earlier PSP studies also suggest that the PSP has such drawbacks as tedious logging tasks, data being error-prone, and a need for tailored adaptation. For example, as students must record their estimated program size, time, and number of defects for each program assignment, the motivation tends to be lowered, and thus leading to a loss of data integrity and authenticity. Besides, a full 2-week training period would be either infeasible or less cost-effective in the industrial arena. Although tailored PSP adaptation may be reinforced, still other consequences like ease of use, discounted effectiveness, instructor loading, and student privacy issues can arise and thus hinder its widespread diffusion [11–14].

While this diffusion puzzle remains unresolved, the more recent PSP studies were also interested in the investigation of disciplined software engineering values and how they might be developed via PSP training [15,16]. Needless to say, these efforts are not easy. Since the PSP is involved with numerous measures and each one of them may occur in only certain phases or aspects, the researchers’
approaches and methodologies used in PSP effect assessment vary tremendously. For example, from a project management approach, researchers tend to focus on estimation accuracy related PSP measures [14,17]. But when taking from the software quality perspective, researchers place more attention on defect related examinations [18–20].

Aiming at resolving these issues, the following research questions can guide and limit the scope of this study:

1. In what way should the PSP issue be approached and exactly what software engineering goals can be reached by PSP training?
2. Which software engineering values and PSP metrics should be used to assess the said PSP effect?
3. How can these measures be assessed accurately and later interpreted with meaningful feedback for PSP users?

To accomplish these goals, the rest of this paper is organized as follows: Section 2 addresses the theoretical framework based on a systematic review of the related literature that might derive an appropriate effect assessment model. Section 3 describes the PSP system and the methodologies used to test this model. Section 4 presents the result of PSP effect. Section 5 discusses the implications that might be derived from the result. Section 6 concludes the research with suggested future studies.

2. Theoretical framework

2.1. PSP and the industrial software engineering standards

Based on the standards like those of capacity maturity model (CMM) and ISO9001 that are used to certify a software company, the PSP can be considered as its equivalent on an individual basis. In [21], Paulk compares ISO9001 with CMM and finds a number of similarities between the two standards. Among them, the bottom line is “to say what you do; to do what you say.” While these requirement definition and documentation work typifies the behavior of a disciplined software engineer in the industry, it could be easily achieved by a training tool like the PSP where it relies on “a well-defined, planned, and measured” process approach [22].

But, what a software company demands on its engineers is a lot more than the baseline discipline. As seen in a longitudinal study of an effective SPI assessment, six factors are considered critical [23]. They include budgeting, scheduling, quality, productivity, customer satisfaction, and employee satisfaction. While the last two measures belong to the long-term goals of TQM and may rely on tools like TSP and CMM, the first four goals can be met by PSP training since it can result in such effects as “better estimation accuracy, improved process and product quality, and increased productivity” [4].

As CMM (initially designed to assess the maturity of software development process) can be used to improve the software process of a large company [24,25], the PSP can also be equally effective in small and median sized business firms [as seen in (26,27)]. Even when IT has evolved rapidly in the last decade, the PSP remains a current and an effective SPI tool in both industrial and academic environments. As seen in [28], both the PSP (a process-based approach) and the agile method (like Extreme Programming, Scrum, Crystal, a people-based approach) can be successfully adopted in the modern world to improve software quality and productivity. Studies in the academia also indicate that the PSP can not only fit in a “traditional” software engineering curriculum [7], but could also be appreciated by the “net” generation when real-world scenarios are incorporated into the course [29].

2.2. PSP life cycle phases in a stage-wise view

As the life cycle phases of PSP (see Fig. 1) evolve from the baseline personal process level to project management, quality management, and the cyclic integration level, the effect of a disciplined software process also occurs on a gradual basis. To better understand the training effect in terms of engineering values, it would be more crystallized by scaling them down into the Plan–Organize, the Track–Manage, and the Review–Optimize stages.

Similar to Hilburn’s term of “organize and plan,” the first (Plan–Organize) stage demands software developers to define, measure, and prioritize target tasks in order to prepare and establish the baseline for later software process improvement [30,22]. These training tasks of estimation, planning, and scheduling, when viewed from the industrial or the cost effectiveness perspectives can foster such engineering values as measurability (or scalability) and predictability (even precision) [11,14,30]. Since creativity may also need to build on discipline and planning can lead to software efficiency and superiority, these disciplined software development values must be trained for each software individual in the initial Plan and Organize stage [2].

In the Track–Manage stage, as Hilburn terms “performance tracking and quality management,” the engineering skills deal not only with the identification of the number, type, and whereabouts of defects, but also with the skill of removing them (i.e. debugging on a program basis) to better manage and control the cost of software quality [30,31]. The engineering values that can be developed from this stage through PSP training, therefore, are traceability and controllability of defects and manageability of software process. In association with the CMM level, these values establish an engineer’s foundation to reaching the second CMM equivalent (the managed) level on a personal basis.

In the Review–Optimize stage, it essentially deals with PSP tasks like design review, code review, and postmortems. The engineering values addressed in this stage are reliability and efficiency. In stead of a product based approach, both of these values are considered in a process based sense with the belief that efficient and superior process and methodology can actually be developed via series of reviews and analyses of PSP templates and reports. When these skills are equipped, engineers can not only deliver software product with quality assurance, but also increase the potentiality of reaching the highest CMM level (the optimized, on an organization basis) [1,32]. Although the terms of “track & review” may often be linked together (just as Plan and Track sometimes are), for the purpose of this study, it is necessary to separate these terms to delineate and help to inspect the PSP effects as they evolve throughout the life cycle phases. While the stage-wise approach encompasses all three stages, those engineering values not directly derived from PSP training (like prior knowledge and skills) or may derive from which only after a successive adoption of it (like persistency) are beyond the scope of this study.

2.3. Pertinent PSP metrics

According to Humphrey and Hayes [3,33], there are only three basic PSP measures: size, time, and defects. All other PSP measures are derived from these basic measures. For the purpose of this study, the terms of PSP metrics (or variables) are hereby defined and categorized with respect to the goals of a disciplined software development process. The detailed list is seen in Table 1.

Within the literature, estimation related PSP effect measures are identified as SEA, TEA, and DEA. Of these measures, SEA refers to the estimation accuracy of program size, TEA to that of program development time, and DEA for defects [3,16,33]. While the estimation accuracy measures serve as indicators of an engineer’s ability to measure and predict software development, his or her ability

to complete a project within the deadline is also critical in PSP effect assessment. Borrowing from Prechelt’s estimation accuracy calculation method [36], a fourth estimation measure APR, meaning the ratio of actual time to planned time, is further added to the list. The inclusion of the APR measure is also consistent with Humphrey’s earned value concept and Boehm’s COCOMO model and GOALS concepts [37,38].

In addition to predictability, measures that depart from the defect perspective including DRR, YLD, and DDS are also identified as crucial in PSP effect assessment. Whereas DRR refers to the ratio of defects removed during the compile and test time, YLD refers to the percent of defects removed before the first compile, and DDS represents total defects removed per thousand lines of code (as defect density) [3,20].

For final PSP effect assessment, LPH (defined as lines of code per hour) and AFR (as prior to post effort ratio) must also be included as pertinent effect measures that depict process review results. As disciplined software engineers must show capabilities of managing defects and processes, such productivity efficiency (LPH) and total quality management (AFR) indicators can best describe if they possess the proficiency in developing a disciplined software product. The references attached on the rightmost column in Table 1 serve to support the above discussion.

Still other factors like individual capability factors of coding, knowledge of software engineering, and the review rate (another PSP measure for lines of code reviewed per hour) are also considered as crucial in PSP effect assessment by the researchers [10,15,34]. However, since these factors are either later found insignificant in SPI assessment or hardly trainable in a single PSP life cycle, they are excluded from the list of this study.

### Table 1

<table>
<thead>
<tr>
<th>Measure</th>
<th>Definition</th>
<th>Implied capability</th>
<th>Engineering value</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEA</td>
<td>Size estimation accuracy</td>
<td>Ability to measure software size</td>
<td>Measurability</td>
<td>[3,9,16,33,34]</td>
</tr>
<tr>
<td>TEA</td>
<td>Time estimation accuracy</td>
<td>Ability to predict development time</td>
<td>Predictability</td>
<td>[3,16,33,34]</td>
</tr>
<tr>
<td>APR</td>
<td>Actual time to plan time ratio</td>
<td>Ability to meet project deadline</td>
<td>Predictability/controllability/traceability</td>
<td>Researcher developed</td>
</tr>
<tr>
<td>DEA</td>
<td>Defect estimation accuracy</td>
<td>Ability to predict defect frequency</td>
<td>Controlability</td>
<td>[3,16,30,33,35]</td>
</tr>
<tr>
<td>DRR</td>
<td>Defect remove rate in compile and test phase</td>
<td>Ability to decrease defect in compile and test</td>
<td>Manageability</td>
<td>[3,16,30,33,35]</td>
</tr>
<tr>
<td>YLD</td>
<td>Yield in percent of defects injected before the first compile that are removed before the first compile</td>
<td>Ability to control defect emergence</td>
<td>Manageability</td>
<td>[3,16,30,33,35]</td>
</tr>
<tr>
<td>DDS</td>
<td>Number of defects removed per thousand lines of code developed</td>
<td>Ability to decrease defect density</td>
<td>Reliability</td>
<td>[3,11,33–35]</td>
</tr>
<tr>
<td>LPH</td>
<td>Total new and changed LOC developed divided by the total development hours</td>
<td>Ability to increase productivity</td>
<td>Efficiency</td>
<td>[3,9,16,30,35]</td>
</tr>
<tr>
<td>AFR</td>
<td>Time spent in design review and code review as a percentage of time spent in compile and test</td>
<td>Ability to manage risk</td>
<td>Reliability/efficiency</td>
<td>[3,8,30,33,35]</td>
</tr>
</tbody>
</table>

### Table 2

<table>
<thead>
<tr>
<th>Stage</th>
<th>Entry level</th>
<th>Intermediate level</th>
<th>Exit level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan</td>
<td>SEA</td>
<td>TEA</td>
<td>APR</td>
</tr>
<tr>
<td>Track</td>
<td>DEA</td>
<td>DRR</td>
<td>YLD</td>
</tr>
<tr>
<td>Review</td>
<td>DDS</td>
<td>LPH</td>
<td>AFR</td>
</tr>
</tbody>
</table>

Based on the previous discussion, a Plan–Track–Review model with the associated PSP effect measures can be organized in Table 2. The cross-wise measures are arranged with respect to the Plan, Track, and Review–optimize categories with varying degrees of engineering value for each stage.

In the Plan stage, PSP tasks including requirement definition, code standard setup, and program size and development time estimation are assessed. The goal of this stage is to improve an engineer’s capability of measurability, predictability and precision...
pertaining to his or her software process development. Therefore, the PSP effect measures that fit to this stage include SEA, TEA, and APR, where SEA can be viewed as an entering assessing measure for planning and APR an exiting measure for this planning stage.

As for the Track stage, though some effect may foster from PSP tasks of coding, compile, and testing, the specific effect examination is focused on credits of defect estimation, identification, and its removal. With this respect, engineering values in this stage are involved with defect traceability and controllability, and process manageability. The associated PSP effect measures are, therefore, DEA, DRR, and YLD.

The Review stage deals essentially with the effects of design review and code review. Since both software quality and productivity must be assessed at this time, the engineering values of process reliability and efficiency are considered, leading to the inclusion of such PSP metrics as DDS, LPH, and AFR. Within this stage, DDS is viewed as an entering variable and an initial assessment measure on overall software quality, LPH on productivity, and AFR on review efficiency (also an exiting measure that concludes all PSP life cycle phases).

In comparison with Hayes’ life cycle segmentation [3], the Plan–Track–Review model is a much simpler one. Literally, this term resembles such terms used in the literature as Plan–Design–Review in [2], Plan–Implementation–Maintenance in [31], and Plan–Track–Analyze in [5]. In essence, our model is much consistent with the life cycle phases described in Boehm’s spiral model. Viewing from the project management field, our model is similar in many ways with Deming’s Plan–Do–Study–Act cycle and a modified version by Hamilton called Plan–Do–Check–Act cycle [39]. In addition, regarding PSP effect assessment, a Plan–Track–Review model application can be helpful in understanding the stage-wise PSP effect and in delineating credits from among trainers, trainees, and PSP as a training tool.

2.5. The assessment methodologies and the criteria reference

Understanding and interpreting PSP training effect is one thing; the deriving of a unified scheme that can accurately assess PSP effect is quite another. Regarding PSP effect assessment, several quandaries need to be overcome. Among which, the first problem is to decide whether we should apply a group or an individual approach and to decide the choice of data ranges. Within the existing literature, the majority of empirical studies rely on the methodologies used by Hayes and Over [3] and Wesslén [4].

According to Hayes, the PSP is designed for individual software engineers and its performance should be examined on an individual basis from level to level [3]. Although group trend can better present the overall effect growth, it is the individual progress that counts. However, since PSP0 through PSP2 phases use aggregate means while PSP3 uses only one assignment data, it would be more logical to apply a comparison of PSP0 with PSP2 than that of PSP0 with PSP3.

The second problem is about the calculation of estimation accuracy related variables that interpret the degrees to which the estimates match the actual results. These variables are involved with all estimation metrics, including program size, process time, and defects. As a general rule of thumb, the estimation calculation method also applies for all of these metrics. However, the methods used for these calculations are quite inconsistent in the literature. For example, as Humphrey [33,37] uses the calculation formula of

\[
\text{Estimation accuracy} = \frac{(\text{Actual} - \text{Planned})}{\text{Planned}}.
\]

Hayes and Over [3] use

\[
\text{Estimation accuracy} = \frac{(\text{Planned} - \text{Actual})}{\text{Planned}}.
\]

Although the second formula gives more intuitive understanding of the signs of overestimation (a plus) and underestimation (a minus), its associated result interpretation should be treated cautiously to avoid a confusion. That is, when Kamatar and Hayes [17] describe their estimation accuracy of 6.7% overestimate, the percentage is a minus in Humphrey’s calculation, but a plus in Hayes’. Still Prechelt and Unger [36] uses a third formula by calculating the quotient of actual and planned measures to obtain his size estimation accuracy value in a percentage ratio form.

The assessing criteria bring a third problem. While it is customary to adopt statistical significances as criteria for assessment, some researchers prefer experiential cuts for PSP effect results. For example, researchers use a greater than 2 for AFR, within an approximation range of ±20%, or simply a group yield (YLD in this paper) of 40%. Among the empirical studies, perhaps Hayes’ Median Improvement Factor method can be adopted best as an improvement reference in that a median refers to a ratio of more than fifty percent and the improvement factor value is explanatory in itself. Based on the literature, a set of tendency criteria for nine PSP effect variables with associated references is given in Table 3. More detailed issues including the result presentation and interpretation are discussed in the following sections.

3. Methodology

3.1. Course setting and data collection

The empirical study was taken during Fall, 2008 with a sample size of 16 graduate students in an 18-week PSP course. Among the subjects, 13 were master level and 3 were doctoral level students. The course was a dedicated PSP course with the objective of training students in establishing software engineering discipline for their future software project development work. Within the 18 weeks, the first 4 weeks covered the baseline PSP concepts, the logging methods, and the code standard setups. The following 10 weeks were used for program development with discussion on topics that occurred from PSP0 thru PSP3. The remaining 4 weeks were used for report generation and feedbacks.

In the PSP course, all students were required to fulfill Humphrey’s 10-assignment programs. As their programming languages, fourteen subjects wrote in Java and two in C++. The assignment interval is about 1 week off each. Since more than two thirds of the students had part-time jobs and almost all had also other course loadings, the study represents a software development scenario in general (though slightly different from the industry) [20,50]. Upon the completion of the course, PSP data were gleaned with variables calculated following Humphrey and Hayes’ formula with only minor modification needed in this study. A survey (based on concepts from [41–43], in that order) was also distributed and its result collected at the end of the course to further understand student reactions on PSP adoption.

To ensure the validity of this study, the students neither had any prior knowledge of this study nor were penalized for their performance. The PSP data used for effect assessment was collected as-is and was organized into three sets to comply with the specific

<table>
<thead>
<tr>
<th>Estimation criteria</th>
<th>Effect variables</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approaching zero</td>
<td>SEA, TEA, DEA</td>
<td>[3,17,30,33,36,40]</td>
</tr>
<tr>
<td>Decreasing</td>
<td>DRR, DDS</td>
<td>[2,3,8,30,33,40]</td>
</tr>
<tr>
<td>Increasing</td>
<td>YLD, LPH</td>
<td>[2,3,17,30,33,40]</td>
</tr>
<tr>
<td>Less than 1</td>
<td>APR</td>
<td>Researcher developed</td>
</tr>
<tr>
<td>Greater than 2</td>
<td>AFR</td>
<td>[3,30,33,40]</td>
</tr>
</tbody>
</table>

testing ranges. The first set was used to detect if an improvement was present covering three PSP phase levels (from PSP0 to PSP2). The second set takes PSP data between programming assignment seven through nine (i.e. 7A–9A as denoted in the literature and for the rest of this paper) to assess those measures occurred only in PSP2. The third is the whole data set based on aggregate mean for all four PSP phases to explore the relationship among PSP metrics.

3.2. Initial improvement assessments

In dealing with PSP effects, most researchers adopt the data as-is and use either a histogram, box-plot, or line chart to display the effect result [2,3,16,30,36,40]. To avoid a biased observation and an erroneous assessment, both mean and median based assessments were used and thus both box-plot charts and line charts were used to describe the distribution and the improvement trends. The rationale is that the box-plot charts can show the distribution and central tendency with medians, but a line chart that uses mean values can better contrast the group effect trend for all measures among the PSP levels.

In terms of the interpretation of the estimation accuracy result, an absolute value form (a fourth formula adding to previous discussion) was employed in this study to calculate estimation accuracy variables. That is, a 300% overestimate (~300%) and a 200% underestimate (~200%) was not totaled to ~100% as in Humphrey’s calculation [3], but as a 500% away from hit. Bearing this in mind, three estimation accuracy variables, namely SEA, TEA, and DEA were converted to their absolute value forms and applied a stricter assessment rule throughout this study. While the absolute value approach is justifiable from the industrial standard, it also helps to assess and interpret the improvement result. Within the literature, this stricter calculation approach was first mentioned in Hayes’ study [3], but actually used by Wesslén [4] in estimation-related assessments.

3.3. t-Test and ANOVA

While the tendency analysis approach better depicts the effect growth through PSP life cycle phases, a series of t-tests can better judge whether the variables meet the SPI criteria with a statistical significance. Since t-test must compare the pre-test and post-test values under the same scheme, most researchers avoid a comparison between two program assignments. Additionally, an ANOVA approach was used to detect if significant improvement existed in any tow adjacent PSP phase levels (e.g. PSP0–PSP1).

As suggested in [44], the phase-level comparison approach with the repeated measures and aggregate means can increase the statistical power of a study. In this study, each PSP effect variable was thus tested except for DEA and AFR which occurred only in PSP2 and PSP3. The range choices for t-test, then, are PSP0 versus PSP2 for seven variables and 7A–9A for two variables (DEA and AFR), both ranges leaving out 10A (or PSP3) as it involves a cyclic integration effect.

Technically, the t-test process can also be replaced by the one-way analysis of variance (ANOVA) method in which more paired comparison details can be displayed with a Post-Hoc test. These details, according to Wesslén [4], can be useful in understanding phase-level improvements and in delineating learning effect from process effect.

3.4. Exploring impacts among stages and PSP metrics

Although the Plan–Track–Review model is derived from a systematic review of the related literature, its fitness can be exemplified by applying PSP data. Since the subjects of this empirical study setting were homogeneous and the only treatment is a full PSP with all 10-assignment, its application result can be meaningful in examining the model fitness. The verification process involves a series of regression and a path analysis. Whereas the regression analysis is used to test if any later stage variable can be accounted for by previous stage variables, a path analysis diagram is used to help visualize the flow and internal effect among PSP metrics. The data collected from these exploration processes can not only be useful in giving clues to the impacts among PSP effect metrics but also worthwhile in interpreting the application results.

Within the process, each later stage variable is tested against all previous stage variables to derive a significant regression model. For example, when DDS is used as a dependent variable, DEA, DRR, YLD, SEA, TEA, and APR are used collectively as independent variables to try out if a significant regression exists. Those significant regressions are then further used to draw a quasi path analysis diagram. Although the purpose varied from those studies which uses path analyses to find an intermediate effect variable [45,46], the method used to derive the result is similar to that of the Partial Least Square method in Green’s study [47].

3.5. The assessment procedure

Finally, a consistent assessment scheme should apply for all nine variables. To exemplify the assessing procedure used in this study, the steps of the simplified and unified procedure are listed as follows:

- **step 1**: choose an interval range for each variable,
- **step 2**: apply proper test and derive test significances,
- **step 3**: compare test result with the expected tendency,
- **step 4**: assess if an improvement result existed based on step 3 and step 2,
- **step 5**: calculate the median improvement factor for each variable,
- **step 6**: interpret the result in engineering values to obtain feedback for each variable, repeat the above steps from 1 through 6 until all variables are assessed.

In step 1, it is preferable to adopt a phase interval range than an assignment range to avoid comparison within small size samples. In step 2, either a t-test or ANOVA is applicable, but the t-test significance results should be retained for later reference. In step 3, the expected tendency refers to the improvement directions that are consistent with the literature. Since all estimation accuracy data are now normalized, the uni-directional tendency comparison rule applies to all nine variables. The real assessment begins in step 4, where both tendency and t-test results need to be weighted. The purpose of the fifth step is to obtain a class-peer reference base, where the median improvement factor stands for the behavior of more than half of the subjects [3]. Additionally, external references based on a larger norm can be used as a contrast with the final assessment result. Finally in step 6, the assessment result can be further interpreted in engineering values (see Table 1) to obtain personal or class strength and weakness information for each PSP effect measure.

A notion must be made is, one must not literally tie the terms of “increasing” and “decreasing” with the variables defined here in describing an assessment result. For example, since those estimation accuracy related variables (SEA, TEA, and DEA) used in this paper are normalized to absolute values, a decreasing trend means a tendency of approaching to zero and is thus considered as an improvement, not as a “decreased accuracy.”
4. Result findings

4.1. Effect distribution and trend analysis result

To exemplify the model application and the feasibility of the simplified assessment procedure, the result of five measures are presented in Figs. 2–6. The rest of the effect measures can also be derived in the same way, but not included here for brevity. In this subsection, two sets of PSP data were used. The first three figures were derived from the data set that covered three PSP phase levels (from PSP0 to PSP2), and the last two (i.e. the result of DEA and AFR effect) used the data set counting from 7A through 9A.

The box-plot chart results on the left side display the distribution tendency with median position marks. The right side of the figures, however, presents the result of group trend calculated by aggregate means. It should be noted that on judging if an improvement existed, both charts must be weighed to avoid a possible biased mean resulting from outliers (displayed as whiskers in box-plot charts).

As indicated in Fig. 2, the improvement of size estimation accuracy is not visible as the group trend “increased” from PSP0 to PSP2. The effort estimation improvement is also minimal since the group trend decreased only slightly in Fig. 3. The improvement of student capability in meeting schedule (APR, or earned value in Humphrey’s term), however, is quite visible as it declined steadily using leverage at PSP1 where PSP process treatment of schedule planning was introduced.

From the box-plot chart side of the three figures, they suggest a relatively stable and consistent behavior of the student progress for all Plan stage training factors but size estimation. As seen on the left side of Figs. 2–4, all boxes are relatively condensed (or slightly dispersed) except for SEA at PSP1 (more loosely dispersed) when size estimation process tool was first introduced and used to improve the guessing result, which could lead to severe project estimation error in an industrial perspective.

Another notion is that all three figures assume a positive Y-axis since the estimation accuracy values were converted to absolute values. The assessment method is thus based on a stricter industrial standard rather than the assessment method used in the literature. Specifically, both program size and coding effort estimation should be approaching the zero, and that the ability ratio of meeting schedule should be approaching one.

In Figs. 5 and 6, since these two measures existed only in PSP2 and PSP3 levels, the tendency evaluation took 7A–9A as interval range (i.e. based on the second data set). As reflected by both box-plot and line charts in Fig. 5, an improvement of defect estimation is highly visible. Similarly, an improvement of the prior to post effort ratio can also be identified in Fig. 6.

In contrasting with the two figures, however, a few interesting phenomena are observed. For example, the boxes of 7A and 9A are smaller for DEA, but their counterparts are larger for AFR. Conversely, the box-plot chart behavior of 8A is quite the opposite to those of 7A and 9A. Finally, we can see that the line chart of Fig. 5 is roughly the reverse to that of Fig. 6.

Although these phenomena represent the result from a single application, they can be logically interpreted by the varieties of coding experience, review rate, or learning effect. Whichever interpretation, it should be noted that the results could not be due to process treatment changes as they all occurred in PSP2, where the design and code review process tool was first introduced. The improvement results, in general, are more or less consistent with those appeared in the existing literature.

4.2. ANOVA results with Post-Hoc tests

Again in this subsection, the first and the second data sets were used to assess the incremental improvement effects. A series of one-way analysis of variance (ANOVA) were used to detect if a statistically significant improvement was existent for all nine variables. The result indicates that only four measures showed such differences and they are given in Table 4. Again, the phase-level testing rule applies for seven variables and the assignment interval range rule applies for two variables.

According to the result of the first three measures in the table, students’ overall defect density, process yield, and defect removal rate had all improved significantly from PSP0 to PSP2. Based on such evidence, it suggests that significant software quality improvement can be achieved through PSP training. The significant defect estimation accuracy improvement result is also consistent with the literature, but under a different assessing range interval from 7A to 9A.

The Post-Hoc test results are not presented here. But according to our analyses, the incremental progresses on software quality are basically consistent with the literature. However, the results of incremental progresses on estimation accuracy are largely inconsistent with the literature since no significance was found even...
in between PSP0 and PSP2. Although it may be too early to draw a conclusion, but this suggests a longer PSP training period may be needed for project planning and estimation.

Within the significant variables, we also find that significant improvement of defect estimation accuracy (DEA) is found only in the 7A–9A pair, but not in two other adjacent pairs. The
complete t-test result for all nine variables with associated significances, however, is provided later in Table 6.

4.3. Regression analysis result

On regression analyses using each later stage variable as a dependent variable and all previous stage variables as independent (or predicting) variables, four significant regressions were found and the result summary is given in Table 5. During the regression analyses, all variable values were counted throughout the four PSP life cycle levels (i.e. using the third data set of this study). Since each observation used in a regression analysis took all variables at a given time, the collective relationship between any two variables in terms of former to later stage predictability becomes meaningful. But, any other forwarding or mixed (inter- or intra-) stage regression would be impractical in analyzing the existence of former stage impact on later stage PSP effect metrics.

As indicated in the table, all four models are significant and that each dependent variable can be statistically accountable by its previous stage variables. That is, the dependent variables can be predicted by using previous stage independent variables. On further inspection of the statistically significant coefficients of predicting variables for each regression model, a series of regression equations are derived as follows:

\[
\begin{align*}
AFR & = 0.009 \times \text{DEA} + 0.004 \times \text{DRR} + 0.407 \\
LPH & = 0.093 \times \text{YLD} + 0.083 \times \text{DRR} + 0.148 \times \text{SEA} + 0.3304 \\
DDS & = 0.162 \times \text{YLD} + 0.997 \times \text{DRR} + 0.221 \times \text{DEA} + 0.4156 \\
DRR & = 0.54656 \times \text{APR} + 0.501 \times \text{SEA} + 0.22242
\end{align*}
\]

From Eq. (3), only DEA and DRR are identified as significant predictors, suggesting that one’s behavior of defect estimation and remove rate can significantly affect his/her prior to post effort ratio. In Eq. (4), it suggests that one’s productivity can be determined by one’s defect yield, defect remove rate, and even size estimation accuracy. This is logical as one’s former period ability to accurately predict project size could also affect his/her later period productivity.

![Fig. 6. Box plot and line charts for appraisal–failure ratio trend.](image)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Interval range</th>
<th>df</th>
<th>( F )</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDS</td>
<td>PSP0–PSP2</td>
<td>(2.45)</td>
<td>9.106</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>YLD</td>
<td>PSP0–PSP2</td>
<td>(2.45)</td>
<td>35.637</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>DRR</td>
<td>PSP0–PSP2</td>
<td>(2.45)</td>
<td>12.634</td>
<td>&lt;0.001</td>
</tr>
<tr>
<td>DEA</td>
<td>AID7–AID9</td>
<td>(2.45)</td>
<td>3.978</td>
<td>0.026</td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th>Dependent variable</th>
<th>Predicting variables</th>
<th>( R^2 )</th>
<th>( p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFR</td>
<td>DEA, DRR, YLD, SEA, TEA, APR</td>
<td>0.332</td>
<td>0.001</td>
</tr>
<tr>
<td>LPH</td>
<td>DEA, DRR, YLD, SEA, TEA, APR</td>
<td>0.319</td>
<td>0.001</td>
</tr>
<tr>
<td>DDS</td>
<td>DEA, DRR, YLD, SEA, TEA, APR</td>
<td>0.130</td>
<td>0.038</td>
</tr>
<tr>
<td>DRR</td>
<td>DEA, DRR, YLD, SEA, TEA, APR</td>
<td>0.349</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Table 6

<table>
<thead>
<tr>
<th>Stage</th>
<th>Variable</th>
<th>Mean-based result (this study)</th>
<th>Median based result and external references</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Progress trend</td>
<td>Significance</td>
</tr>
<tr>
<td>Plan</td>
<td>SEA</td>
<td>13.2 → 20.5</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>TEA</td>
<td>29.8 → 26.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>APR</td>
<td>1.16 → 1.01</td>
<td>–</td>
</tr>
<tr>
<td>Track</td>
<td>DEA</td>
<td>63.6 → 25.0</td>
<td>.010*</td>
</tr>
<tr>
<td></td>
<td>DRR</td>
<td>119.2 → 43.6</td>
<td>&lt;0.001**</td>
</tr>
<tr>
<td></td>
<td>YLD</td>
<td>5.7 → 47.9</td>
<td>&lt;0.001**</td>
</tr>
<tr>
<td>Review</td>
<td>DDS</td>
<td>121.8 → 70.9</td>
<td>.015*</td>
</tr>
<tr>
<td></td>
<td>LPH</td>
<td>23.1 → 24.9</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>AFR</td>
<td>1.12 → 1.15</td>
<td>–</td>
</tr>
</tbody>
</table>

* Stands for \( p < .05 \).
** Stands for \( p < .001 \).
In Eq. (5), all three attributes in the Track stage (but none from the Plan stage) have a significant impact on DDS, suggesting the overall quality effect may be strongly based on one’s Track stage performance. The sixth equation is an interesting one since the result indicates that the defect remove rate is predictable by both the size and an alternate time estimation measure, which lends support to a possible carry over effect from the Plan stage to the Track stage. However, the small accountability ($R^2 = 0.130$) also gives clues that a wide range of other factors including individual knowledge and experience \[34,49\] are yet missing in our Plan stage.

4.4. The path analysis diagram

To further understand the meaning of the regression analysis result, a quasi path analysis diagram is drawn in Fig. 7. For clarity’s sake, all paths drawn here bear significance and those not are discarded. Each $R^2$ labeled underneath its associated effect variable represents a goodness of fit. The asterisks along with the path coefficients indicate their significance levels in which one asterisk represents for $p < 0.05$, two asterisks for $p < 0.01$ and three for $p < 0.001$.

As indicated by the path coefficients in the diagram, when DRR decreases, DDS will decrease as well, but LPH and AFR will increase. This is consistent with the literature in that as the defect ratio of compile-test phase gets smaller, the overall defect density will also decrease, and in turn this promotes the productivity and consequently the prior to post effort ratio tends to go high.

Within the diagram, the most robust regression model can be seen in the DDS pattern, to which three contributing variables (DEA, DRR, and YLD) together can interpret the total variance to 94.9%. Although there remain a dangling TEA and some intuitively paradoxical paths (e.g. from SEA to LPH), the diagram itself can roughly reflect the dynamics of the stage pattern and the effect relationships that lie within.

4.5. PSP effect result summary

The overall PSP effect result summary is given in Table 6, which includes both mean and median based assessment results. The mean portion is comprised of student progress trend and its improvement with statistical significance (if any). The median based improvements are based on their median improvement factors with both internal (i.e. this study) and external (Wesslén’s in 2000, and Hayes’ in 1997) references.

As indicated in the report, software quality-related PSP effect is obvious as all four variables (DEA, DRR, YLD, DDS) matched the SPI criteria with statistical significance. But other improvements are less obvious as they either did not improve or improved with no statistical significance. The improvement results of two other sub-quality factors extracting from the DRR measure, namely the DRRC and DRRT (as indicators of process quality and product quality respectively \[3,4\]), are not included. The mean results of these measures are the same with the two studies, and the median counterpart is also more or less comparable with the external reference results.

Viewing from the median based assessment result in the table, more than 50% of the students improved with a median improvement factor greater than one for all effect measures but SEA (0.67, as indicated in parentheses). Although this result neither supports nor denies the commonly accepted PSP training effect, the result suggests that at least half of the students improved on eight out of nine software engineering goals through this training.

Weighing from the Plan–Track–Review perspective, the results suggest, in general, that the students performed well in the Track stage, only moderately in the Review stage, and poorly in the Plan stage. When these results are further interpreted in terms of their specific engineering values (as shown in Table 1), they imply that a few disciplined software development effects can actually be achieved from PSP training. For example, a software practitioner can become more skillful in defect control and software process management as disclosed by the DRR and YLD indicators in Table 6. Secondly, a practitioner can also ameliorate his or her defect predictability (DEA) and software product reliability (DDS) to a certain extent. Lastly, there is also a possibility that one might develop one’s abilities to predict project time, to meet project schedule, and to improve software productivity as seen in the results of TEA, APR, and LPH.

As for the abilities to estimate project size (SEA) or to manage overall project risk (AFR), however, no positive PSP training merits can be implied from this study. In further contrast with external references, the results tend to be more agreeable with those of

![Quasi path analysis diagram of variable flow.](image)
Wesslén’s (a similar graduate level academic setting), but less so with those of Hayes’ (an industrial setting).

5. Discussion

5.1. Threats to validity

5.1.1. Threats to internal validity

When using PSP data as an empirical study, four types of validity threats must be considered: namely, the conclusion, internal, construct, and external validity threats [50]. The first two of these threats are data related validity issues. As mentioned by Johnson and Disney, about 5% of PSP data may be defective, and less than ten percent of them are identified as logging errors [51]. Other than logging errors, outliers (whiskers) can also skew the statistical result if either totally discarded or blindly included [52]. The worst kind is a possible data corruption resulting from “intentional” data errors, which may make the results look better [50].

To avoid these threats, the course instructor (one of the authors of this study, also a certified PSP trainer) was cautious and promised that he would not flunk any student as long as he or she studied hard and honestly worked through the ten assignments. The truth is, all PSP data as well as the survey result were collected upon the completion of the course, and the students neither had any prior knowledge of this study, nor were penalized for poor performance (i.e. all passed the course). On each assignment submission, however, the course instructor was strict in checking if any calculation, omission, or transfer error existed in the assignment. To ensure data validity, a data inconsistency check was performed by essentially counting the total number of defects injected with the number removed as suggested in [52]. Since no data consistency errors were identified for all 160 assignments, the internal validity is sound in this study.

In [10,52], Paulk has also discussed in great details on the outlier issue. The outliers can be identified in a number of ways, including the popular inter-quartile method which calculates the upper (Q3) and lower (Q1) inter-quartile limits. The outliers are also easily visualized in a box-plot chart (as seen in Figs. 2–6). Although outliers can seriously influence the mean value and thus the estimate, the median counterpart is less likely affected. Due to the limited sample size used in this study, a series of backup data sets that exclude the outliers were used for cross-examinations, but not for the result assessment. It is also the same reason that both mean and median assessment schemes (and approaches) are used throughout this paper.

5.1.2. Threats to external validity

Typically, a large sample size can increase the precision of an estimate. But this is neither justified mathematically nor considered as a sole factor for a sampling choice. That is, the homogeneity of the subjects can also increase the representativeness of our sample. Moreover, when the assessment result is used to compare and contrast with a much stronger reference base (for example n = 298 in [3] and n = 131 in [4]), the size of a small pool becomes less critical. Pedagogically speaking, since a part of our purpose is to provide useful feedback for an individual learner against one’s peers, perhaps the meaning of an internal peer comparison based on a smaller pool appears to be more critical than an external reference based on a large sample size.

Other than the descriptive statistical analyses portion, it should also be noted that the sample size used in the regression and path analyses is actually 64 and not just 16 (the number of the subjects). For model validation and the generalization of the results, however, the small sample size is still an issue. Further studies and more evidence are needed.

5.2. Implications of the inconsistent estimation results

The insignificant estimation accuracy results (i.e. SEA, TEA) reflected in this study could be due to a number of reasons and the following discussion may offer some explanations. At the beginning, we had suspected that they might be due to the absolute value form. But a set of confirmatory analyses denied this assumption. The outlier cause will not stand either since the result did not show any difference when the backup data sets (those excluding the outliers via the inter-quartile method) were tested. The chance of data logging error is also minimal since most students reflected that they were capable of logging and confident about their logging integrity (see survey questions 2–4).

While the estimation effect results of the study disagree with most studies in the existing literature, they are also partially supported by some studies under similar settings. For example, other researchers also found an insignificant effort estimation improvement [43], and an insignificant size estimation improvement [8,54].

Although there might also be a PROBE reliability issue (mentioned in [55]), we suspect that the inconsistency is mainly due to that our students lacked real-world experience (as compared to Wesslén’s study) and that a significant effect could not be achieved through a short period of time (i.e. requiring a longer learning curve) for planning and estimation training. Conversely, the significant PSP effect results on software quality may suggest that both process and product quality can be improved via PSP training within a relatively shorter period of time.

Whereas it is too early to draw conclusion from this single application, our goal aims to look for further evidence to fully interpret the result and to explore other underlying factors and implications.

5.3. Observations from a PSP course instructor

After teaching software engineering topics for many years, we believe that the so-called “discipline” needs to be trained in order to meet the challenges of contemporary software development tasks [56]. This can also be disclosed by the fact that students kept on coming back for re-training on industrial requests. While also deserving the blame, we suspect that there is no silver bullet (see [57]) and the institutionalization of such discipline could not be established without a successive tool adoption, not even with a tool like the PSP. Other observations along with implications of the survey result (see Appendix A) are given below.

First of all, the PROBE method does cause a problem for most PSP learners, even for a graduate student. This can be reflected by that the course instructor who spent a great deal of time in teaching and assisting the students to improve their logging errors. The survey result also indicates that although most students appreciated the PSP, they decided not to use such technologies as size estimates and effort logging in the future. This further points out the pending need for a modified logging tool for the PSP (like LEAP or HackyStat as suggested in [131]). There is also evidence that these more convenient tools are already adopted academically and proven effective, to name a few [58–60].

The students also expressed that they particularly liked the coding template and the postmortem part of the PSP, from which they could benefit valuable information of their personal weaknesses (e.g. types of errors). As seen in the survey result, the majority of students (more than eighty percent) had perceived the importance of the code review process and was committed to adopt it in the future. This information also serves to support our opinion on the inclusion of the AFR measure as a crucial indicator of software process quality and efficiency.

As for a feasible PSP curriculum design, it is recommended to arrange it at the graduate than an undergraduate level (as justified
in [61]. The rationale behind is that an undergraduate student is normally lack of coding (and much less project) experience. It is also recommended that the assignments of a PSP course can be tailored to imitate the real-world software projects. While the already tight curriculum restricts the length of a full PSP, a partial adoption of PSP in many other computer courses can also be effective if only that the tedious logging load and the programming difficulties could be ameliorated (as seen in [58,62]). An experience report on PSP application in database courses and a list of suggestions to instructors by Humphrey serve to agree with these points [11,63].

5.4. Industrial application

When applying our assessment model to the industry, a few issues need to be considered. For instance, a full length PSP is hardly feasible and the program assignments definitely need to be tailored to meet the industrial needs. Ideally, a project comprising a set of assignments that resemble typical challenges of a software company should be designed for PSP training. In industry, a practitioner also needs to learn programming on the fly in order to meet their tight project schedule. The real coding discipline should be emphasized on skills of using external library, not on coding from the scratch or on effort in struggling with irrelevant assignment contents (i.e. the mathematical problems offered by Humphrey). The number of assignments and the length of training can also be lessened to fit the software company. Whichever case, the assignments should cover all PSP phases (or the Plan–Track–Review stages) in order to foster the said PSP training effects.

Another drawback with the PSP is that it does not provide an acceptance test mechanism for the “end products” (assignments). In our study the acceptance test was conducted by the students themselves. But in the real world this is done by the real customer. Alternatively, an external “customer proxy” could help in counting the defects (in an extended sense) in the final outcome against the requirements posed at the beginning. If this customer proxy could not be onsite throughout the project period, he/she should at least be involved two to four times to ensure that the team builds the right thing (like the method used in Honig’s study [29]).

As for the assessment of a practitioner’s PSP training effect, we recommend to concentrate on those exiting metrics (see Table 2) in the Plan–Track–Review model. For example, as the industrial standard aims at meeting project schedule in the Plan stage, the APR might be a better indicator than TEA. A training officer can also apply flexible weights on different metrics whenever he/she deems appropriate. Finally, since the PSP is for a continuous improvement of personal software process, it is recommended to arrange PSP training regularly and continuously to ensure that a practitioner can meet both essential and accidental software challenges.

6. Conclusion

In this article, we have proposed an assessment model to test against the commonly accepted PSP training effects. As demonstrated by the empirical data collected from a PSP course, we have applied the model in the classroom. The assessment results of student performance regarding PSP training effect are presented with our interpretation and discussion. A PSP effect result summary is also included to provide feedback for students as well as educators based on industrial, academic, and class-peer standards.

Although some of the assessment techniques are not very much different from the existing literature, the new viewpoints and concepts incorporated in this study can increase understanding of the results of applying the PSP. Unlike other similar studies, this paper has taken a Plan–Track–Review approach, in which nine PSP metrics are identified as critical and pertinent to software engineering values. Within the proposed assessment model, an initial regression result also indicates the possibility that former stage effect may have an impact on later stage PSP metrics.

While the long-term and short-term PSP training effects are unclear and other possible factors are missing in this study, our future work aims to refine the assessment model and to clarify some of these concerns. Finally, we regret that the already tight higher education curriculum has hindered the PSP diffusion, and thus limited the sample size in recent empirical PSP studies like this one. However, we strongly advocate replications of similar studies to ensure that our students are truly prepared for their future software development challenges.

Acknowledgements

The authors would like to express thanks to those students who diligently studied the course and unselfishly provided the data in this study. The authors would also like to thank the anonymous reviewers whose most insightful comments are much helpful to the improvement of an earlier version.

Appendix A. Survey result

See Tables 7 and 8.

Table 7

<table>
<thead>
<tr>
<th>Question</th>
<th>1 (%)</th>
<th>2 (%)</th>
<th>3 (%)</th>
<th>4 (%)</th>
<th>5 (%)</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 I understand the PSP topics covered in class and used in the logs</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>73.33</td>
<td>26.67</td>
<td>4.3</td>
</tr>
<tr>
<td>2 I am capable of completing the PSP logs</td>
<td>0</td>
<td>0</td>
<td>13.33</td>
<td>73.33</td>
<td>13.33</td>
<td>4.0</td>
</tr>
<tr>
<td>3 I devoted sufficient time to completing the PSP logs</td>
<td>0</td>
<td>0</td>
<td>40</td>
<td>40</td>
<td>20</td>
<td>3.8</td>
</tr>
<tr>
<td>4 The information recorded in my PSP logs is accurate</td>
<td>0</td>
<td>0</td>
<td>33.33</td>
<td>53.33</td>
<td>13.33</td>
<td>3.8</td>
</tr>
<tr>
<td>5 The PSP topics covered were helpful for doing quality work in the course</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>33.33</td>
<td>66.67</td>
<td>4.7</td>
</tr>
<tr>
<td>6 I can see that the PSP topics covered would be helpful for doing quality work in future jobs</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>46.67</td>
<td>53.33</td>
<td>4.5</td>
</tr>
<tr>
<td>7 I plan to use the PSP concept in future programming work</td>
<td>0</td>
<td>0</td>
<td>13.33</td>
<td>60</td>
<td>26.67</td>
<td>4.1</td>
</tr>
</tbody>
</table>

References


