

# An Experience Report of Applying the Personal Software Process Methodology

**Wen-Hsiang Shen**

Department of Information Management  
The Overseas Chinese Institute of Technology  
Taichung, Taiwan, R. O. C.

**Nien-Lin Hsueh and Peng-Hua Chu**

Department of Information Engineering  
and Computer Science Feng Chia University  
Taichung, Taiwan, R. O. C.

***Abstract** - The purpose of this study is to investigate whether PSP can be adopted to foster pedagogical effect in an academic class. In this study we examined five personal process improvement areas of the PSP: size and effort estimation accuracy, product quality, process quality, and personal productivity. Surveys were taken at the end of this course to understand students' attitudes toward the Personal Software Process. This paper explores the results of these effects and surveys.*

**Keywords:** Personal software process, Process improvement, Quality management

## 1. Introduction

The Personal Software Process (PSP) is used by software engineers to gather and analyze data about their work, and to produce empirically based evidence for the improvement of planning and quality in future projects. Related research [1,3,7-10] have suggested that adopting the PSP results in improved size and time estimation and in reduced numbers of defects found in the compile and test phases of development. However, the results of most studies evaluating the PSP are based on analysis of PSP data – the PSP lets students collect their own data to show how they've improved in this research. In PSP model [6], project life-cycle activities are divided into sequential phases: planning, design, design review, code, code review, compile, test, and postmortem.

The seven process levels [2] used to introduce the PSP are shown in Figure 1. Each level builds on the prior level by adding a few process steps to it. Each new level introduces new elements and more complicated material until the engineers reach the highest level, PSP 3.

- *PSP 0.* In the first level, this level covers how to record development time and how to log each compile and test defect.
- *PSP 0.1.* This level adds size measurement and the process improvement proposal, a form that engineers use to record the process problems they encounter as well as their ideas for addressing them.
- *PSP 1.* In this level, engineers are introduced to the PROBE method, which uses historical data to estimate size and determine the accuracy of the estimate. PROBE is a regression-based size-estimating method developed specifically for PSP.
- *PSP 1.1.* This level adds resource and schedule estimating and earned-value tracking. Earned-value tracking allows them to weight the relative importance of each task and to judge their progress as they finish some tasks early and others late.

- *PSP 2*. This level introduces design and code reviews, as well as quality measurement and evaluation. Using defect data from their earlier exercises, engineers also develop personal design and code review checklists.
- *PSP 2.1*. In this level, engineers learn design specification techniques and ways to prevent defects.
- *PSP 3*. In this highest process level, this level covers design verification techniques and methods for adapting PSP to engineers' working environments.

Each PSP level introduces new measures to help engineers manage and improve their performance. These measures are derived from the three basic PSP measures: development time, defects, and size. By applying the techniques and procedures learned during the course, engineers see how to apply PSP to large-scale software development.

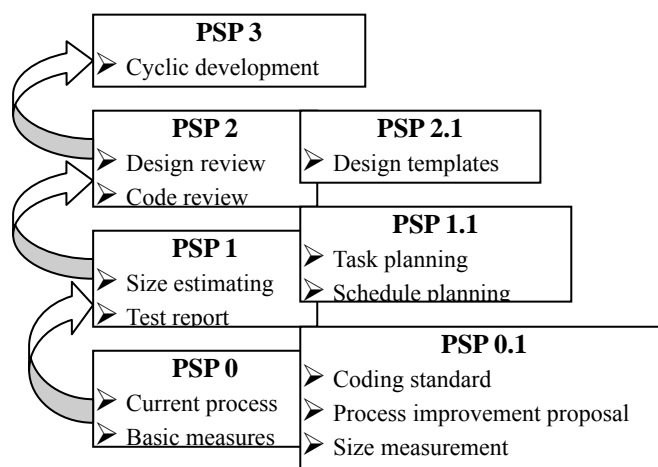


Figure 1. PSP Process Levels

In this research, we intend to analyze the student's work records in an academic PSP course from five areas: size estimation accuracy, effort estimation accuracy, product quality, process quality and personal productivity. The rest of the paper is organized as follows. The case study design is described in section 2, and the experimental result is analyzed and reported in section 3. In section 4, we illustrate and discuss the surveys results. Conclusion is in the last section.

## 2. Case Study Design

Successful application of PSP requires training. PSP can be studied in a one-semester graduate-level university course, which includes a PSP textbook and 10 programming and five analysis exercises [4]. PSP is now being taught at more than 20 universities in the US and at institutions in Europe, South America, and Australia. Some universities, too, now offer an introductory freshman course that is designed to start engineers off on the right track [5].

This case study began by teaching a one semester graduate-level university course on the PSP, modified in certain ways in an attempt to improve the quality of PSP data. The students in the course submitted all of their paper forms to the instructor after each assignment. Some errors required students to correct and resubmit prior forms. The set of paper PSP forms collected over the course of the semester comprises the original PSP dataset, and can be thought of as one experimental treatment. In all, a PSP class consisting of 16 students provide the data used

in this report. Students use statistical methods during PSP training to analyze their personal data and make judgments about the benefits of changes to their personal software processes. In this report, we perform a secondary analysis of these students' data in aggregate form.

### 3. Study Results

In this study we examine five personal process improvement areas of the PSP: size and effort estimation accuracy, product quality, process quality, and personal productivity. This study focuses on group trend. The result shows that students improve their capabilities in the first four areas, but do not have a significant improvement in the last area (productivity). The detail analysis is described in the following sub-sections.

#### 3.1 Size Estimation

The PSP provides a proxy-based estimation method to help students decompose the program and estimate the size of each element, based on historical data. Introduction of this method is designed to enable students to become more exact estimators of their own work. The distributions shown in Figure 2 illustrate the performance of students' size estimation for the three PSP levels. The values plotted were derived by summing the data for the three assignments in each PSP level and computing the estimation accuracy value. The horizontal axis in Figure 2 represents the value of estimation accuracy, ranging from -100% to +250%. The vertical axis represents the number of estimations for each assignment at a particular level of estimation accuracy. With respect to the performance of the group, the number of engineers achieving a value near zero during PSP level 2 is more than twice that of PSP level 0. Besides, the long 'tail' of the distribution in the PSP Level 2 (stretching out to +250%) is significantly reduced by the PSP Level 3.

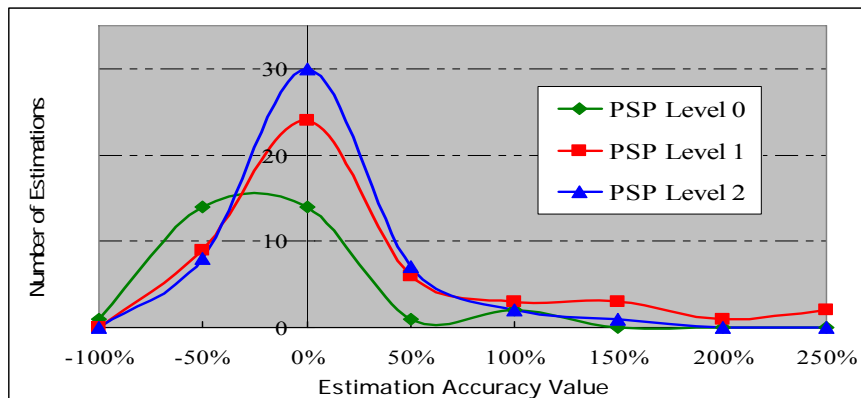


Figure 2. Distributions of Size Estimation Accuracy by PSP Level

#### 3.2 Effort Estimation

Use of chronological data for deriving effort estimates is ordinary practice in the software industry now. However, estimation at the level of an individual engineer's workload remains a challenge. The PSP training provides engineers with the capacity to make estimates, and to progress the estimating process, at the level of an individual engineer. This ability is clearly demonstrated in the results presented here. A comparison of the three distributions in Figure 3 shows that the majority of students overestimating their effort in PSP level 0-1. For PSP

level 2, the distribution is more nearly symmetrical (the number of engineers overestimating is closer to the number of engineers underestimating). Finally, for PSP level 2, the distribution is closer still to the desired shape (symmetrical with narrow range, centered on zero), and the number of estimations with estimation accuracy values near zero is substantially larger.

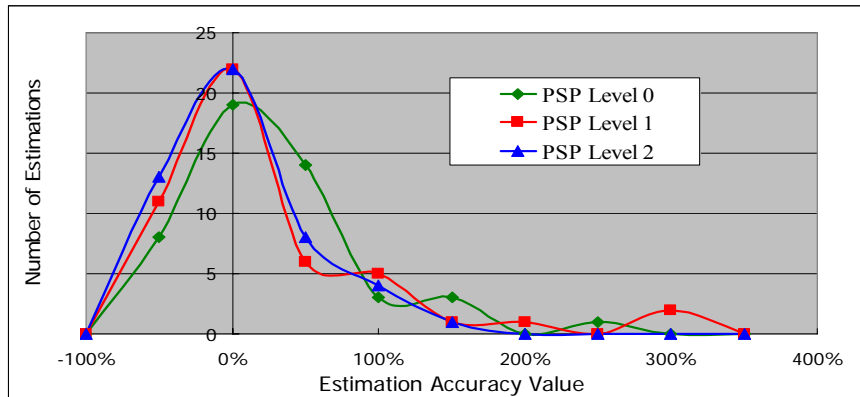


Figure 3. Distributions of Effort Estimation Accuracy by PSP Level

### 3.3 Defect Density

Defect counts and measures of defect density have usually served as software quality measures. The PSP uses this method of measuring product quality, as well as several process quality metrics. Figure 4 delineates the change in average defects per KLOC removed during compile and test, as well as defects per KLOC over the entire life cycle. As Figure shows, the PSP level 1 and PSP level 2 is a significant decline in defect density as PSP level 0. More importantly, the number of defects removed in the compile and test phases (taken together) are substantially lower for PSP level 2, as compared to PSP level 0. Figure 5 provides a more detailed examination of changes in defect density. The horizontal axis represents the number of defects per KLOC and the vertical axis represents the number of students reporting defect densities at a given value. The data plotted are pooled defect densities for each PSP level. The three curves show defect densities in the compile and test phase.

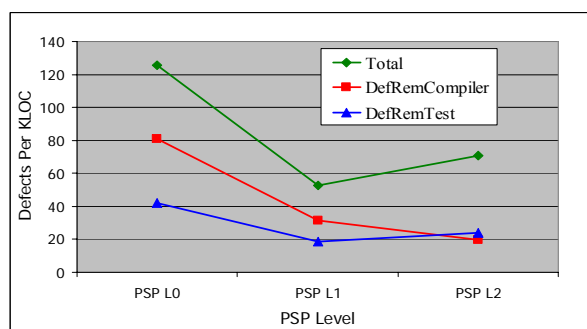


Figure 4: Trends in Average Defect Density

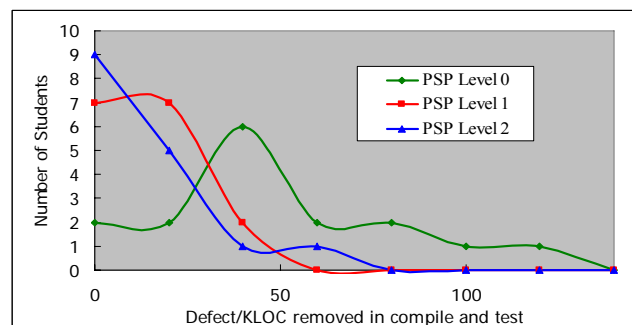


Figure 5: Defect Density Distributions for Compiler and Test Phase

### 3.4 Defect Yield

Yield is critical factors affecting personal software processes [10]. Reducing test defects is an important software development objective because having many test defects implies poor software quality, and because

test defects are relatively expensive to fix. Yield is the percentage of defects injected before the compile phase that are removed before the first compile. The PSP training teaches engineers to examine process quality by quantifying the yield of their personal software process. In general, the goal is to work for a yield of 100%. Figure 6 shows the trend in average yield across the three PSP levels, with the dotted lines representing  $\pm 1$  standard deviation. The increase in average yield from approximately 0.5% to nearly 42% from PSP level 1 to PSP level 2 is significant changes. The major process change that occurs between PSP levels 1 and 2 is the introduction of formal design reviews and code reviews. A more detailed depiction of the changing values of yield for each assignment is provided in Figure 7. As the figure illustrates, prior to assignment 7, very few students removed defects from their programs before the first compile. Starting with assignment 7, however, more than 80% of the students removed more than 27% of the defects before they compiled the program.

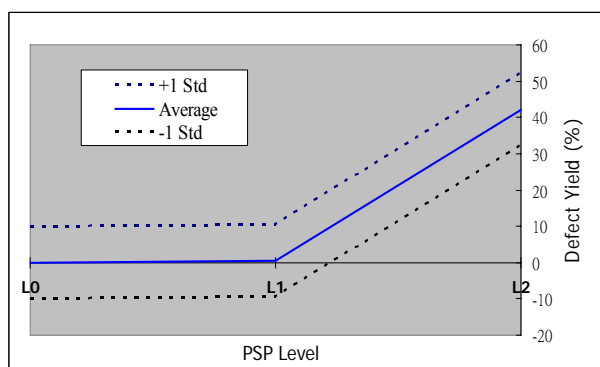


Figure 6: Average Yield

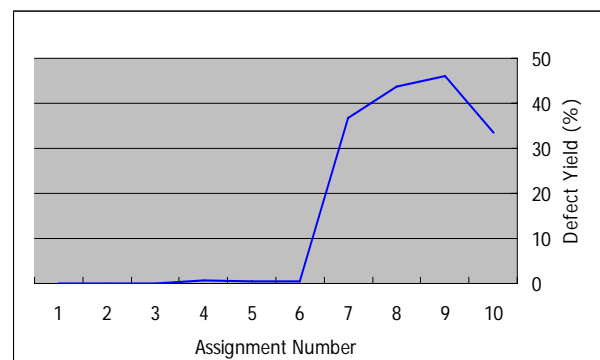


Figure 7: Average Yields for Each Assignment

### 3.5 Productivity

Productivity is a major focus of most organizations that produce goods for customers. In PSP training, the data collected by the students allow them to compute lines of code per hour (LOC/Hr) as a measure of their personal productivity. The trend value shows unobvious variation in average productivity among the three PSP levels. The mean values near 24.4 LOC/Hr for all three levels. Examining the averages, we see that there is an increase in productivity of approximately four lines of code per hour between levels 0 and 1. Between levels 1 and 2, there is a reduction in productivity of approximately two line of code per hour. Finally, from level 0 to level 2, there is an increase in productivity of approximately three line of code per hour.

## 4. Results of Student Surveys

This course's review consisted of the nine questions, the former seven questions, which the students ranked on a 5-point scale from strongly agree to strongly disagree. There was also a "No Opinion" option, the last two questions have multiple option (Size measurement after development, Effort (time) log during development, postmortem after development, Size estimation, Effort (time) estimation, Task planning and schedule planning, Code review, Design review, Design using design template). The nine questions, listed in Table 1, tried to determine whether the students understood the PSP logs, completed the logs accurately, and recognized any future trend in applying the Personal Software Process.

Table 1. End-of-course Survey Questions

1. I understand the PSP topics covered in class and used in the logs
2. I am capable of completing the PSP logs.
3. I devoted sufficient time to completing the PSP logs.
4. The information recorded in my PSP logs is accurate.
5. The PSP topics we covered were helpful for doing quality work in the course.
6. I can see that the PSP topics we covered would be helpful for doing quality work in future jobs.
7. I plan to use the PSP concept in future programming work.
8. After the PSP training, which technologies (or practice) are most important and helpful for you?
9. After the PSP training, which technologies (or practice) you will apply in your future development?

The results of questionnaire for the nine questions are given in Tables 2 and 3. Multiplying the number of responses by the response number and dividing by the total number of respondents gives the average displayed in the last column. Hence, a low value indicates a positive response to the question, and a value above 2.5 indicates a negative response to the question. In general, students felt they understood the PSP topics (Question 1) and were capable of completing the logs (Question 2), although they weren't highly convinced of this. A majority admit they did not spend adequate time on the logs (Question 3). Around half also indicate the data in their logs is not quite accurate (Question 4). No matter how, There is seldom a much higher negative response to the others questions, indicating the students are aware of the significance of the Personal Software Process. Besides, from the results of question 8 and 9 (see Table 3, option 1-3), we can find an interesting incident: most students think that size measurement, effort logging and postmortem are important for software development, but they do not plan to do these activities in their future development (for example, 86.67% students think postmortem after development is important, but only 33.3% will that again). This may result from that we do not have an automatic tool to support these activities. Thus, tool support is very important for PSP methodology. In the meanwhile, Table 3 option 7-8 indicates the activities of code review and design review is most people would like to implement. However, reviews are quite useful for eliminating most of the defects found in compile, and many of the defects found in test. But to substantially reduce test defects, better quality designs are needed.

Table 2. Survey Results of Question 1-7

Question Number	Response						Average
	Strongly Agree 1	2	3	4	Strongly Disagree 5	No Opinion	
1	26.67%	73.33%	0%	0%	0%	0%	1.7
2	13.33%	73.33%	13.33%	0%	0%	0%	2.0
3	20.0%	40.0%	40.0%	0%	0%	0%	2.2
4	13.33%	53.33%	33.33%	0%	0%	0%	2.2
5	66.67%	33.33%	0%	0%	0%	0%	1.3
6	53.33%	46.67%	0%	0%	0%	0%	1.5
7	26.67%	60.0%	13.33%	0%	0%	0%	1.9

Table 3. Survey Results of question 8-9

Option	Question Number	
	8	9
Size measurement after development	60.0%	20.0%
Effort (Time) log during development	73.33%	33.33%
Postmortem after development	86.67%	33.33%
Size estimation	73.33%	53.33%
Effort (time) estimation	73.33%	66.67%
Task planning and schedule planning	80.0%	66.67%
Code review	86.67%	80.0%
Design review	80.0%	73.33%
Design using design template	73.33%	60.0%

## 5. Conclusion

As the effectiveness of PSP becomes evident by taking a PSP course, the students can be trained how to manage and estimate their time and how to do well-quality work. Additionally, students build a tangible foundation for learning about design and development methods. When students learn the PSP, they understand the importance of process, and they learn how to define, plan, measure, and control their work. Based upon these measurements, the students can track their productivity, make better predictions for future projects, and gain insight to what types of errors they make, and learn how to remove defects earlier in their development process. Still more, the greater part favorable comments seemed to come from those students who have worked, and recognize the benefits of such methods. It is likely that this topic will be continued from this course in future semesters.

## 6. References

- [1] G. C. Green et al., The impacts of quality and productivity perceptions on the use of software process improvement innovations, *Information and Software Technology*, Vol.47, 2005, pp.543–553.
- [2] P. Ferguson et al., Results of Applying the Personal Software Process, *Computer*, Vol. 30, No. 5, May 1997, pp.24–31.
- [3] W. Hayes, J. Over, The personal software process (PSP): an empirical study of the impact of PSP on individual engineers, Technical Report SEI-97-TR-001, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 1997.
- [4] W. S. Humphrey, *A Discipline for Software Engineering*, Addison-Wesley, New York, 1995.
- [5] W.S. Humphrey, *Introduction to the Personal Software Process*, Addison-Wesley, Reading, Mass., 1997.
- [6] P. M. Johnson and Anne M. Disney, A Critical Analysis of PSP Data Quality: Results from a Case Study, *Journal of Empirical Software Engineering*, December, 1999.
- [7] J. Kamatar W. Hayes, An Experience Report on the Personal Software Process, *IEEE Software*, Nov/Dec 2000 pp.85-89 Volume: 17, Issue: 6.
- [8] M. Morisio, Applying the PSP in Industry, *IEEE Software*, 17(6), 2000, pp.90-95.
- [9] L. Prechelt, B. Unger, An Experiment Measuring the Effects of Personal Software Process (PSP) Training, *IEEE Transactions on Software Engineering*, Vol. 27, No. 5, May 2001, pp.465-472.
- [10] Z. Xiaoming et al., Critical factors affecting personal software processes, *IEEE Software*, Nov./Dec., 2000, pp.76-83.