

The Personal Software Process in the Classroom: Student Reactions (An Experience Report)

Susan K. Lisack

*Purdue University, Computer Information Systems Department
W. Lafayette, IN 47907
sklisack@tech.purdue.edu*

Abstract

Most people would agree that the quality of a product is related to the quality of the process used to develop that product. The quality of a computer program is often measured by the number of defects it contains. The Personal Software Process was developed to help programmers measure and improve their personal productivity. A subset of the Personal Software Process has been suggested as appropriate for beginning college students in introductory programming courses. This subset has been used over the past two years in several sections of a large first and second semester programming course, with mixed success. Students recorded data during the development of their lab assignments and major programming project, and submitted these along with their programs. Surveys were taken at the end of each course to determine student attitudes toward the Personal Software Process. Many students failed to recognize the benefits of such a process, and felt that it just took time away from learning the programming language. This paper explores the results of these surveys.

1. Introduction

Computer applications frequently contain defects that result in a lower quality product. Gibson[2] cites data that shows that even software engineers with experience can inject around 100 defects into their code for each 1000 lines of code written. A quarter of these defects may remain in the product until final system testing. At this stage it can take many hours to track down the source of the problem and correct it.

The Personal Software Process provides disciplined practices that a software engineer can use to measure and improve personal productivity, with the goal of producing high-quality software, on schedule and within budget. It was developed at the Software Engineering Institute at Carnegie Mellon University. The full Personal Software Process (PSP) is intended to be taught in a full-semester, graduate-level programming course. A simplified version of this process, geared towards freshmen and sophomores, can be taught using the book "Introduction to the Personal Software Process" by Watts S. Humphrey[4]. It is designed as a supplementary text for a programming course with short, easy-to-read chapters. Ideally the material is spread across a two-semester programming sequence, which provides more opportunity to apply the process to programming assignments.

In the Fall 1997 semester, the first half of the simplified PSP process was taught in an introductory programming course for Computer Information System majors. This same group

of students completed the simplified PSP materials in the second programming course during the Spring 1998 semester. Subsequently, the Personal Software Process was taught solely in the second programming course in Fall 1998 and Spring 1999. All of these courses are taught to over a hundred students at one time. In addition to meeting in lecture twice a week, the students also meet once a week in a lab setting in smaller groups of around twenty.

2. Student Characteristics

Students for these particular courses come into the programming course sequence after completing several introductory courses on general information systems concepts, computer architecture, PC applications, and PC databases. In the PC applications course the students gain skills in Windows, file manipulation, word processing, spreadsheets and presentation tools. In the PC database course they learn SQL in Access. Besides programming, these students will go on to take courses in networking and telecommunications, systems analysis and design, database, and various electives in these and other areas. The students can choose between two main program areas for their degree – a networking and telecommunications area, or the information systems area. Both paths require the programming courses. However, many of the students in this major do not aspire to be programmers (about 30% take jobs as Application Developers).

3. Using the Personal Software Process in the Classroom

The Personal Software Process itself involves completing a variety of logs to track time spent in various categories of the software development process, the defects found in programs and what stages they were found and removed. Some of this data is then combined on other forms to evaluate trends. The number of forms to be filled out can seem overwhelming, both to a new instructor of the process and to a student who is also just learning how to program. For this initial trial of teaching the Personal Software Process, three logs (the Time Log, Job Number Log and Defect Log) were selected as the forms the students would fill out during programming assignments. Copies of the various forms were provided with the book in electronic versions (as Word documents).

During the first semester, students maintained time and defect data in the Word versions of these forms, and they were periodically checked during lab sessions. In the next semester, the Job Number Log was converted to an Excel spreadsheet and formulas were added to automatically calculate the rate (minutes per lines of code) and accumulate the “to-date” data from multiple projects. The students were asked to electronically submit the three log files at the completion of several lab assignments and a larger project. Maintaining three separate log files became cumbersome, and several students realized they could combine them onto multiple sheets within the same spreadsheet file. This improvement is available to students in the current semester. It is also much easier for the instructor to manage one file from each student when they are electronically submitted.

One problem with incorporating PSP into an existing course is that the same programming topics still need to be covered in approximately the same depth, and any time spent in class on PSP takes away from time previously spent on programming topics. Several methods were tried to teach the Personal Software Process while still covering the same amount of programming material. One semester, students read a chapter a week, and the professor summarized the key concepts. After a few weeks, the students maintained the selected logs. A small percentage of

the two exams tested the key concepts. A significant number of students did not even purchase the book that semester.

The next year, the PSP material was only taught in the second programming course. Students were assigned two chapters per week and asked to outline the key topics. During class, students were randomly called on to list the key concepts and answer questions. In the current semester, there was an early quiz over the first chapters. This has increased the number of students who obtain and look at the book, but has not improved their attitudes toward the material.

4. Mistakes in Completing Student Logs

Although the Humphrey book chapters are relatively short and easy-to-read, a major problem encountered has been that a good number of students do not complete the forms correctly. Disney, et al.[1] reported on problems in both industrial and academic settings with the PSP data collection and analysis. In their analyses of data from ten students over nine projects, they discovered the students made over 1500 errors.

In evaluating the logs submitted by our students from the Spring, 1999 class, only ten of 25 students sampled submitted usable data for all four assignments. In the Time Log, some students failed to calculate the total time spent on the program, or did not subtract off their interruption times. The Units column, which for a program represents the lines of code written or changed, was frequently empty or incorrect. Several students reported the times in hours instead of minutes, in spite of the fact that there was a key at the top of the form requesting time in minutes and units as lines of code.

Table 1. Types of PSP Log Errors Observed

Log Type	Error Type	# Students out of 25
Time Log	No delta (total) time computed	2
	Units is reported as 1, not lines of code	2
	Units is delta time, not lines of code	2
	Completion and/or Units columns are empty	5
	Delta time includes interruption time	1
	Wrong categories indicated (e.g. Review for test and debug)	1
	Units reported in hours instead of minutes	4
Job Number Log	Actual time & units are not filled in	1
	Units is reported as 1, not lines of code	1
	Units are small numbers, not lines of code	2
	Program time includes course lecture time	1
	Multiple entries for one job	4
	Design time not included in project time	2
Time Log & Job Number Log	Units reported in hours instead of minutes	4
	Reported times don't match between the two logs	4
	Time & Job Using hours instead of minutes	4
Defect Log	Defects not separated by job	3
	Wrong project heading at top of file	1

Several of the same errors appeared in the Job Number Log, including the reporting of hours instead of minutes, and not showing lines of code for the units. A number of students made multiple entries for one program in the Job Number Log instead of combining the times and units into one row. When comparing times between these two logs, the data often did not match. In some cases, it looked like the discrepancy was the failure to include program design time as part of the time spent on the program. These common errors from the 25 students sampled are listed in Table 1.

When the Job Number Log was converted to spreadsheet form with built-in equations, and made available to all students, a number of students failed to enter their data in the correct cells, thus causing calculation errors. To correct this, some of these students then deleted the formulas and hand-calculated their results or left them empty.

In looking at the submitted log files, another problem appeared to be under-reporting of defects. Of 57 log files sampled, a total of 534 defects were reported. This is in contrast to Disney, et al.[1], who reported that by the middle of their course, each student typically records at least 100 defects. In our sample, one student reported 132 defects the next highest student reported 75 defects, and the third highest had 40 defects. There was a low of 11 total defects reported for the three lab assignments and a larger mid-term project.

5. Results of Student Surveys

Why do students fail to complete the logs correctly? There are a number of possible reasons, including failure to read the book, not understanding the forms, not caring whether the figures are accurate, or disliking the entire PSP process. This latter result was supported by surveys conducted at the end of the Fall 1997, Spring 1998 and Spring 1999 semesters, which revealed that the students strongly disliked the Personal Software Process and questioned its value in the introductory programming courses.

Disney, et al.[1] reported that the PSP data collection seems time consuming and disruptive, and the time to record certain defects exceeds the time to correct them. We also heard these same complaints from the students.

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 concepts in future programming work.
8. The PSP topic should continue to be a part of this course, in some form.

Figure 1. End-of-course Survey Questions

Each semester’s survey consisted of the same eight questions, which the students ranked on a 6-point scale from strongly agree to strongly disagree. There was also a “No Opinion” option. The eight questions, listed in Figure 1, tried to determine whether the students understood the PSP logs, completed the logs accurately, and recognized any future value in applying the Personal Software Process.

The survey results for each of the three semesters are given in Tables 2, 3 and 4. In comparing the survey results, you should note that the Fall 1997 semester results are from students in the first programming course, who completed the first nine (of 20) chapters of the PSP book. The Spring 1998 semester results are from students in the second programming course who completed the remaining chapters of the PSP text. For the most part, these were the same students as from the Fall 1997 semester. The surveys from the Spring 1999 semester were completed by students from the second programming course. These students completed the entire PSP text in one semester.

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 3.5 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 enough time on the logs (Question 3). Around half also indicate the data in their logs is not accurate (Question 4). There is a much higher negative response to the last four questions, indicating the students do not recognize the value of the Personal Software Process.

Table 2. Fall 1997 Survey Results (from 34 students)

Question Number	Response							Average
	Strongly Agree 1	2	3	4	5	Strongly Disagree 6	No Opinion	
1	17.6%	23.5%	14.7%	20.6%	8.8%	8.8%	5.9%	3.1
2	32.4%	14.7%	26.5%	11.8%	5.9%	5.9%	2.9%	2.6
3	14.7%	8.8%	11.8%	23.5%	20.6%	17.6%	2.9%	3.8
4	17.6%	23.5%	20.6%	11.8%	11.8%	11.8%	2.9%	3.1
5	11.8%	0.0%	8.8%	11.8%	26.5%	38.2%	2.9%	4.6
6	11.8%	11.8%	8.8%	20.6%	8.8%	32.4%	5.9%	4.1
7	5.9%	5.9%	17.6%	11.8%	11.8%	41.2%	5.9%	4.5
8	11.8%	0.0%	2.9%	14.7%	26.5%	38.2%	5.9%	4.7

Table 3. Spring 1998 Survey Results (from 42 students)

Question Number	Response							Average
	Strongly Agree 1	2	3	4	5	Strongly Disagree 6	No Opinion	
1	4.8%	19.0%	23.8%	26.2%	7.1%	19.0%	0.0%	3.7
2	16.7%	28.6%	28.6%	11.9%	4.8%	9.5%	0.0%	2.9
3	4.8%	7.1%	14.3%	9.5%	26.2%	35.7%	2.4%	4.6
4	9.5%	14.3%	23.8%	19.0%	7.1%	26.2%	0.0%	3.8
5	2.4%	2.4%	9.5%	9.5%	21.4%	52.4%	2.4%	5.1
6	4.8%	19.0%	11.9%	16.7%	16.7%	26.2%	4.8%	4.1
7	4.8%	9.5%	9.5%	11.9%	11.9%	47.6%	4.8%	4.7
8	7.1%	4.8%	2.4%	19.0%	16.7%	47.6%	2.4%	4.8

Table 4. Spring 1999 Survey Results (from 40 students)

Question Number	Response							Average
	Strongly Agree 1	2	3	4	5	Strongly Disagree 6	No Opinion	
1	12.5	37.5	20.0	12.5	7.5	5.0	5.0	2.8
2	20.0	40.0	22.5	10.0	7.5	0.0	0.0	2.5
3	2.5	20.0	7.5	10.0	32.5	27.5	0.0	4.3
4	12.5	15.0	22.5	10.0	15.0	22.5	2.5	3.7
5	0.0	0.0%	10.0	5.0	25.0	57.5	2.5	5.3
6	7.5	15.0	12.5	17.5	25.0	20.0	2.5	4.0
7	0.0	2.5	10.0	12.5	22.5	45.0	7.5	5.1
8	5.0	2.5	10.0	10.0	20.0	45.0	7.5	4.9

6. Conclusions

The goal of the Personal Software Process, to offer methods to increase the product quality and productivity of a software engineer, is a good one. However, first and second-year college students may not be ready to appreciate its benefits. This paper reports on the results of trying to teach the Personal Software Process to several large groups of students in introductory programming courses. Since the students were already struggling with learning algorithms and language syntax, the vast majority of the students felt that the logs required by the Personal Software Process took up too much additional time to be taken seriously. They made numerous

errors in completing the forms, and exited the course with a bad attitude toward the process. The few favorable comments seemed to come from older students who have worked, and recognize the benefits of such methods. It is likely that this topic will be discontinued from these courses in future semesters.

For the current semester, it may be helpful to go over the log forms again and review common errors after the students have completed one set of logs. In the future, the PSP process could be taught in an upper-division elective programming course (where presumably the students are more interested in programming) or at the graduate level.

7. References

- [1] Disney, A. and Johnson, P. (1998). Investigating Data Quality Problems in the PSP. *SIGSOFT '98*: 143-152.
- [2] Gibson, R. (1997). Applied Software Process Improvement. *Proceedings of the Americas Conference on Information Systems*: 596-598.
- [3] Grove, R. (1998). Using the Personal Software Process to Motivate Programming Practices. *SIGCSE Bulletin*: 98-101.
- [4] Humphrey, W. (1997) *Introduction to the Personal Software Process*. Reading, MA: Addison-Wesley.