7 A Cross Course Analysis of Product Quality Improvement with PSP

Fernanda Grazioli, Universidad de la República
William Nichols

7.1 Introduction and Background

These days, more and more businesses develop, combine, and include software in their products in different ways. Companies need to develop software to support the design, manufacture, or delivery of the products and services they provide. Therefore, although they might not realize it, all businesses are becoming software businesses. As the software component of their business grows, schedule delays, cost overruns, and quality problems caused by software become their main business problems. This is why, despite their best management efforts, companies find their risk of failure increasing along with the increase in the size or complexity of the software they produce.

Software products are made of hundreds to millions of lines of code, each one handcrafted by a software engineer. Software businesses depend on people, so their technical practices and experience strongly influence the outcome of the development process.

The Personal Software Process (PSP) is a defined and measured software process designed to be used by an individual software engineer. The PSP directly addresses software business needs by improving the technical practices and individual abilities of software engineers, and by providing a quantitative basis for managing the development process. By improving individual performance, PSP can improve the performance of the organization.

For many years, the Software Engineering Institute (SEI) has trained software engineers in PSP. Over that period, the course format has changed twice. Several versions of the course use the same exercises, but introduce process steps in modified sequences. An earlier version of the course has several published studies demonstrating improvement in developer performance with process insertion, but the retrospective analysis left some threats to external validity [Paulk 2006; Hayes 1997; Wohlin 1998; Rombach 2008; Kemerer 2009; Paulk 2010]. One threat is the confounding of process insertion with the gaining of domain experience as related programs are developed. A related threat is that observations might alter the subject performance as in the Hawthorne effect [Mayo 1949]. Moreover, there are not yet studies about how the latest two course versions are working, nor are there studies about how different approaches to introducing process correlate with performance and quality results in PSP courses. The PSP community and the SEI need to know how effectively these courses work. The academic and industrial communities need assurance that the process can be taught effectively and that the process insertion would have positive and substantial benefits.

Given this situation, the objective of this study is to use the PSP data from the latest two course formats to determine whether reviews and design improve product quality, or if such improvement is only a consequence of gaining experience in the problem domain. We measure quality as the quantity of defects found per KLOC in Unit Testing. Defect counts and measures of

defect density (i.e., defects per KLOC) have traditionally served as software quality measures. The PSP uses this method of measuring product quality as well as several process quality metrics. The consequence of high defect density in software engineering is typically seen in the form of bug-fixing or rework effort incurred on projects. Typical defect densities of delivered products range from one to five defects/KLOC [Davis 2003].

7.1.1 Concept Introduction on PSP Courses

The PSP courses incorporate what has been called a "self-convincing" learning strategy that uses data from the engineer's own performance to improve learning and motivate use. The last two course versions introduce the PSP practices in steps corresponding to six PSP process levels. The older version name is "PSP for Engineers I/II (PSPI/II)" and the latest version name is "PSP Fundamentals and Advanced (PSP Fund/Adv)."

Each level builds on the developed capabilities and historical data gathered in the previous level. Engineers learn to use the PSP by writing seven or eight programs (depending on the course version), and by preparing written reports. Engineers may use any design method or programming language in which they are fluent. The programs typically contain around one hundred lines of code (LOC) and require a few hours on average to be completed. While writing the programs, engineers gather process data that are summarized and analyzed during a postmortem phase. There are three basic measures in the PSP: development time, defects, and size. All other PSP measures are derived from these three basic measures.

During the course, the students were given eight or seven exercises (eight in PSPI/II and seven in PSP Fund/Adv), which were mainly programs for statistical calculations. PSP has a maturity framework that shows its progression on improvement phases, also called levels. Students completed their exercises while following the process attained at each PSP level.

The PSP levels introduce the following set of practices incrementally:

- PSP0: Description of the current software process, basic collection of time and defect data
- PSP0.1: Definition of a coding standard, basic technique to measure size, basic technique to collect process improvement proposals
- PSP1: Techniques to estimate size and effort, documentation of test results
- PSP1.1: Task planning and schedule planning
- PSP2: Techniques to review code and design
- PSP2.1: Introduction of design templates

Table 16 shows which PSP level is applied on each program assignment, for each course version.

Table 16: PSP Levels for each Program Assignment

Program Assignment	PSP Fund/Adv	PSP I/II
1	PSP 0	PSP 0
2	PSP 1	PSP 0.1
3	PSP 2	PSP 1
4	PSP 2	PSP 1.1
5	PSP 2.1	PSP 2

Program Assignment	PSP Fund/Adv	PSP I/II
6	PSP 2.1	PSP 2.1
7	PSP 2.1	PSP 2.1
8	-----	PSP 2.1

7.2 Data Set and Statistical Model

We used data from the eight-program course version, PSPI/II, taught between June 2006 and June 2010. Additionally, we used data from the seven-program course version, of PSP Fund/Adv, taught between December 2007 and September 2010. These courses were taught by the SEI at Carnegie Mellon University or by SEI partners, by a number of different instructors in multiple countries.

We began with 347 subjects in total, 169 from the PSP Fund/Adv course and 178 from the PSPI/II course. From this we made several cuts and ran data-cleaning algorithms to include only the students who had completed all programming exercises, in order to clean and remove errors and questionable data.

To determine the cuts on the data set, we first developed an integrated data storage model. We designed that model to support the analysis and the assessment of data quality, based on the data quality theory. Data quality is an investigation area that has generated a great workload in the last years and it is mainly focused on defining the aspects of data quality [Batini 2006; Lee 2002; Neely 2005; Strong 1997] and on proposing techniques, methods and methodologies to the measurement and treatment of the data quality [Batini 2006; Lee 2002;Wang 1995].

The first step toward identifying quality problems was to understand the reality and context to be analyzed. This includes the Personal Software Process in itself [Humphrey 1995], exploring the tool for recording data and the model of the database, in addition to the Grading Checklists used by instructors for the correction of exercises performed during the course. Afterwards, we analyzed the dimensions and quality factors proposed by Batini and Scannapieco [Batini 2006] to this set of data, which are interesting to measure and consider. In this way, we thoroughly identified and defined possible quality problems that the data under study might contain, we implemented the algorithms required for cleaning and collecting the metadata, and finally, we executed those algorithms. Major data quality problems were related to the consistency, accuracy, completeness, and uniqueness dimensions. This meant that following that data quality process, our data set was reduced to 93 subjects in total, 45 from the PSP Fund/Adv course and 47 from the PSPI/II course.

Differences in performance between engineers are typically the greatest source of variability in software engineering research, and this study is no exception. However, the design of the PSP training class and the standardization of each engineer's measurement practice allow the use of statistical models that are well suited for dealing with the variation among engineers.

In the summarized analyses presented, we studied the changes in engineers' data over seven programming assignments. Rather than analyzing changes in group averages, this study focuses on the average changes of individual engineers. Some engineers performed better than others from the first assignment, and some improved faster than others during the training course. To

discover the pattern of improvement in the presence of these natural differences between engineers, the statistical method known as the repeated measures analysis of variance (ANOVA) is used [Tabachnick 1989]. In brief, the repeated measures analysis of variance takes advantage of situations where the same people are measured over a succession of trials. By treating previous trials as baselines, the differences in measures across trials (rather than the measures themselves) are analyzed to uncover trends across the data. This allows for differences among baselines to be factored out of the analysis. In addition, the different rates of improvement between people can be viewed more clearly. If the majority of people change substantially (relative to their own baselines), the statistical test will reveal this pattern. If only a few people improve in performance, the statistical test is not likely to suggest a statistically significant difference, no matter how large the improvement of these few people.

7.3 Analysis and Results

First, we define the following variables and terms:

- Subject –student who performs a complete PSP course.
- Course Type –a PSP course version. It can be PSP Fund/Adv or PSPI/II.
- Program Assignment or Program Number –an exercise that a student has performed during the PSP course. It can be 1, 2, 3, 4, 5, 6, 7 or 8.
- PSP Level –one of the six process levels used to introduce the PSP in these course versions. It can be PSP0, PSP0.1, PSP1, PSP1.1, PSP2, or PSP2.1. Each program assignment has a corresponding PSP level according to the PSP course version. Since we wanted to analyze the introduction of concepts during the courses, we group PSP0 and PSP0.1, we group PSP1.0 and PSP1.1, and we analyze PSP2.0 and PSP2.1 separately. So the PSP Level variable can be seen in plots as 0, 1, 2, or 2.1, respectively.
- Defect Density in Unit Testing (DDUT) – 1000 · Total defects removed in testing / Actual added and modified LOC

As we stated in the introduction, the objective of this study is to use the PSP data from the latest two course formats to demonstrate whether reviews and design improve the quality of a product in test. And we use defect density as a measure of quality, so we define defect density in unit testing as the independent variable in our analyses. Figure 34 shows a bar-whisker chart of DDUT grouped by course type and PSP level. Figure 35 shows a bar-whisker chart of DDUT, but in this case grouped by course type and program assignment. These charts are descriptive and allow us to get a clearer idea of the defect density behavior.

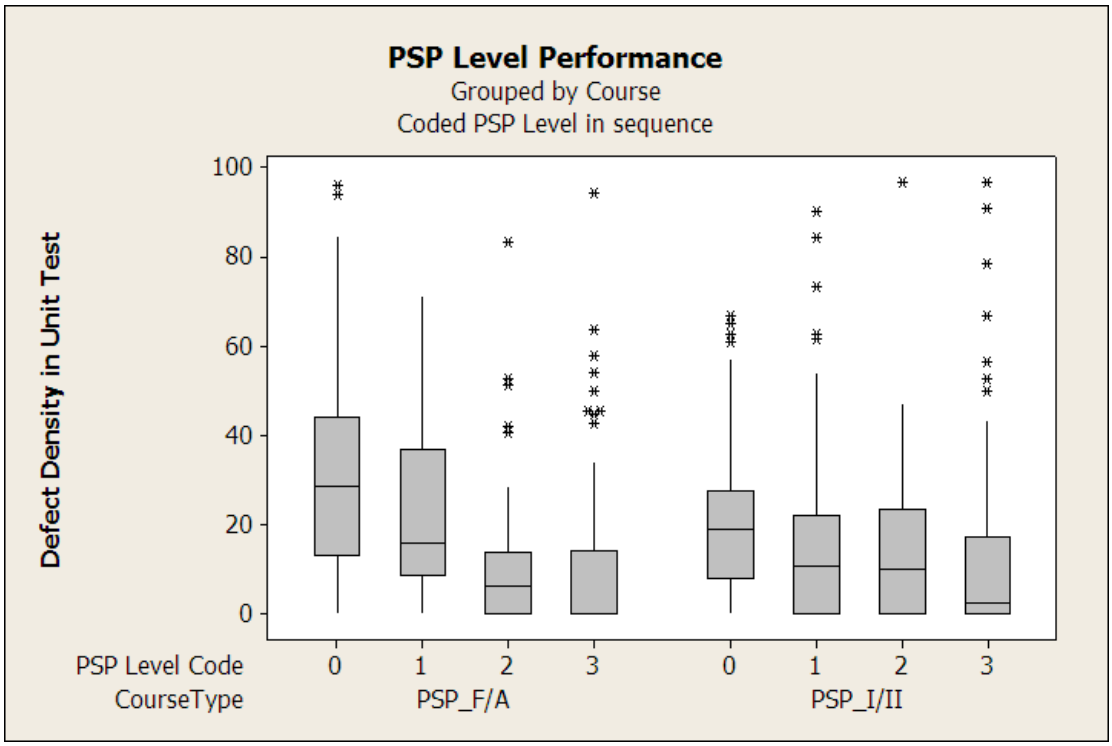

Figure 34: Defect Density in Unit Testing grouped by Course Type and PSP Level

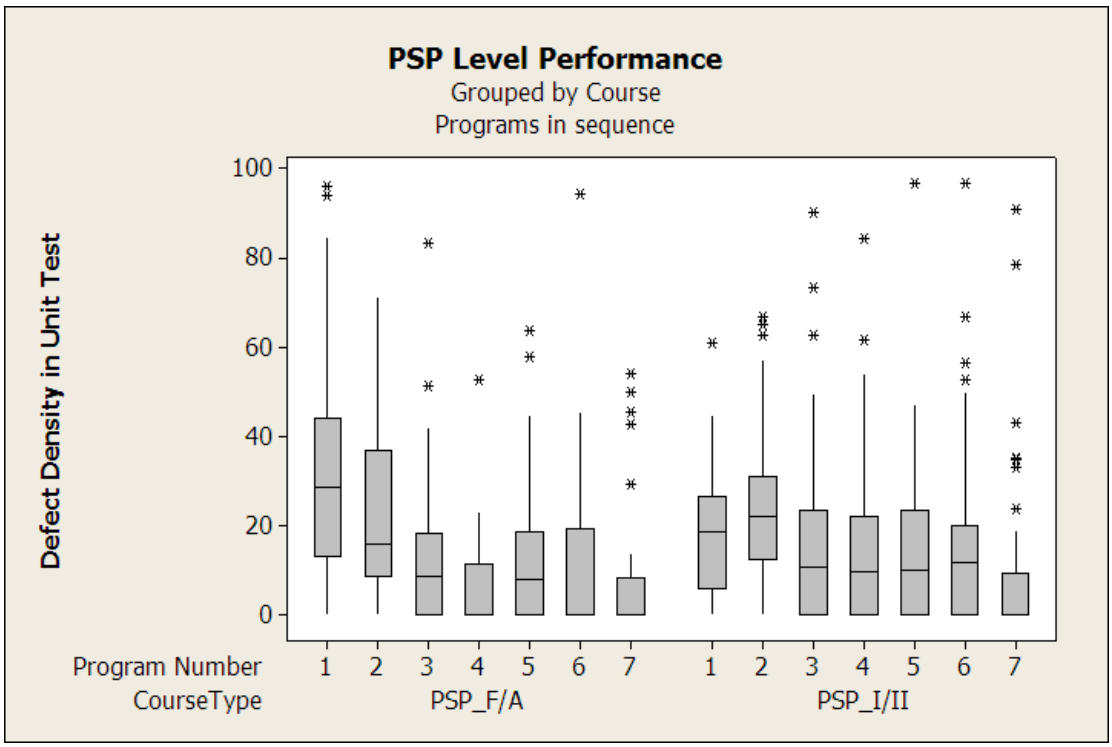

Figure 35: Defect Density in Unit Testing Grouped by Course Type and Program Assignment

To reach that objective, we considered a direct way through an ANCOVA analysis, using PSPI/II and PSP Fund/Adv as categorical values to segment the data into two parts: the program assignment as the independent variable and the PSP level as the hidden variable. But we could not

come to a conclusion because the correlation between the PSP level and the program assignment was strong. Also, using ordinal program numbers in ANCOVA may not be a sound approach. Correlations between the factors are a strong anti-indication for ANCOVA [Tabachnick 1989]. Therefore, we decided to create a more indirect procedure and analyze the results based on specific differences between the two courses using the PSP level.

We next developed an indirect procedure to examine relationships between program number, PSP level, and performance in the data. It consists of three steps, each one based on an ANOVA analysis using defect density in unit testing as the independent variable.

Before running any ANOVA, we checked to see if the data satisfy all assumptions for the correct use. The only assumption that could not be fully satisfied was the normality assumption. We transformed the data into normal form using different techniques as suggested in [Tabachnick 1989], and the one that worked better was a log transformation and adding a constant to the zero observations. Nonetheless, some deviations from normality persisted at very low defect levels. Recent works recommend not to log-transform count data [O'Hara 2010] for two reasons. First, low counts cause distortions. Second, such transformation typically does not affect the results if the data is unimodal. The data satisfies the unimodal mounded condition, and the fact that transformed and untransformed PSP data provides comparable results has been previously observed with [Hayes 1997]. We did our analysis considering both transformed and untransformed data.

The first step of the analysis procedure consisted of performing a series of one-way ANOVA between each program number using course as the grouping factor. This establishes whether the assignments have different DDUT means between the courses. If there are not different means between courses, then the analysis is done because the program-by-program results for the two courses do not differ. We ran seven tests, one for each program number and, since we found significant differences, we proceeded to the next step.

The second step of the analysis procedure consisted of performing two-way (repeated) ANOVA. Separate, repeated ANOVA analyses grouping by program number were performed for each course type. We applied the two-tailed significance test at 0.05, which is equivalent to a 0.025 significance level for a one-tailed test. Table 17 and Table 18 summarize the ANOVA discussed above for this step for the courses PSP Fund/Adv and PSP I/II, respectively. A third two-way ANOVA grouping by course and program number was performed on the combined data, and the results are summarized in Table 19.

Both the separate course data and the combined data showed a general downward trend in defect level with program number, irrespective of process level. This was as we expected based on the correlation between PSP level and program number. We found no statistical significance between consecutive programs for either the PSP Fund/Adv or the PSP I/II. However, for the combined data set, we found significant reduction in defect levels between programs 2 and 3, and 3, and 4. The significance in the combined set results from the increased sample size.

In the PSP Fund/Adv course we found that there is significance between Program 1 and Programs 3, 4, 5, 6, and 7. We interpret that this shows improvements between PSP0 and PSP2 or PSP2.1 (depending on which program from 3 to 7). In regard to PSP I/II, we found that there is

significance between Program 1 and Program 7, and this is consistent with improvements between PSP0 and PSP2.1.

We also found that there is significance in PSP Fund/Adv between Program 2 and Program 3, 4, 5, 6, and 7. This makes sense because it shows improvement between PSP1 and PSP2 or PSP2.1 (depending on which program from 3 to 7). In regard to PSP I/II, we found that there is significance between Program 2 and Program 4, 5, 6, 7. This also makes sense because it shows improvement between PSP0 and PSP1 or PSP2 or PSP2.1 (also depending on which program from 4 to 7).

Table 17: ANOVA Outputs for Program Assignment Comparison in PSP Fund/Adv

PSP Fund/Adv				
Program Assignment (I)	**Program Assignment (J)**	**PSP Level**	**Mean difference (I-J)**	**Sig.**
1	2	1	*,154*	*,999*
	3	2	*1,364*	*,003*
	4	2	*2,348*	*,000*
	5	2.1	*1,602*	*,000*
	6	2.1	*2,139*	*,000*
	7	2.1	*2,347*	*,000*
2	3	2	*1,210*	*,012*
	4	2	*2,193*	*,000*
	5	2.1	*1,448*	*,001*
	6	2.1	*1,985*	*,000*
	7	2.1	*2,192*	*,000*
3	4	2	*,983*	*,082*
	5	2.1	*,237*	*,994*
	6	2.1	*,774*	*,301*
	7	2.1	*,982*	*,082*
4	5	2.1	*-,745*	*,347*
	6	2.1	*-,208*	*,997*
	7	2.1	*-,001*	*1,000*
5	6	2.1	*,537*	*,731*
	7	2.1	*,744*	*,349*
6	7	2.1	*,207*	*,997*

Table 18: ANOVA Outputs for Program Assignment Comparison in PSP I/II

PSP I/II				
Program Assignment (I)	**Program Assignment (J)**	**PSP Level**	**Mean difference (I-J)**	**Sig.**
1	2	0.1	*-,485*	*,820*
	3	1	*,502*	*,795*
	4	1.1	*,730*	*,381*
	5	2	*,816*	*,248*
	6	2.1	*,948*	*,109*
	7	2.1	*1,517*	*,001*
2	3	1	*,987*	*,083*
	4	1.1	*1,216*	*,012*
	5	2	*1,301*	*,005*
	6	2.1	*1,434*	*,001*

PSP I/II				
Program Assignment (I)	**Program Assignment (J)**	**PSP Level**	**Mean difference (I-J)**	**Sig.**
	7	2.1	*2,003*	*,000*
3	4	1.1	*,228*	*,995*
	5	2	*,314*	*,975*
	6	2.1	*,446*	*,871*
	7	2.1	*1,015*	*,067*
4	5	2	*,0854*	*1,000*
	6	2.1	*,2179*	*,996*
	7	2.1	*,786*	*,290*
5	6	2.1	*,132*	*1,000*
	7	2.1	*,701*	*,433*
6	7	2.1	*,569*	*,682*

Table 19 shows a summary of the ANOVA for the combined data.

Table 19: ANOVA Outputs for Program Assignment Comparison Combined Course Data

Program Assignment (I)	**Program Assignment (J)**	**Mean difference (I-J)**	**Sig.**
1	2	*-,166*	*,509*
	3	*,933*	*,000*
	4	*1,539*	*,000*
	5	*1,209*	*,000*
	6	*1,544*	*,000*
	7	*1,932*	*,000*
2	3	*1,099*	*,000*
	4	*1,705*	*,000*
	5	*1,375*	*,000*
	6	*1,710*	*,000*
	7	*2,098*	*,000*
3	4	*,606*	*,016*
	5	*,276*	*,271*
	6	*,611*	*,015*
	7	*,999*	*,000*
4	5	*-,330*	*,188*
	6	*,005*	*,985*
	7	*,393*	*,117*
5	6	*,335*	*,182*
	7	*,723*	*,004*
6	7	*,388*	*,122*

As a summary of this step, we can say that for each course we found significant difference only between assignments with different PSP levels. For the combined course we found a significant difference between programs 2 and 3 (introduced in PSP level 2 in PSP Fundamentals) and programs 3 and 4 (PSP the second use of level 2 in Fundamentals and PSP level 1.0 to 1.1 estimation by parts, in PSP I).

According to the design and review techniques introduced in the corresponding PSP levels, we expected these improvements. Figure 36 shows the estimated marginal means of the log-

transformation of defect density in unit testing versus program number, for both courses. The graphic shows how the two courses perform differently. The declining defect level is more consistent and larger in PSP Fundamentals through the introduction of PSP 2.0. Defect levels appear to be more consistent by the end of the courses.

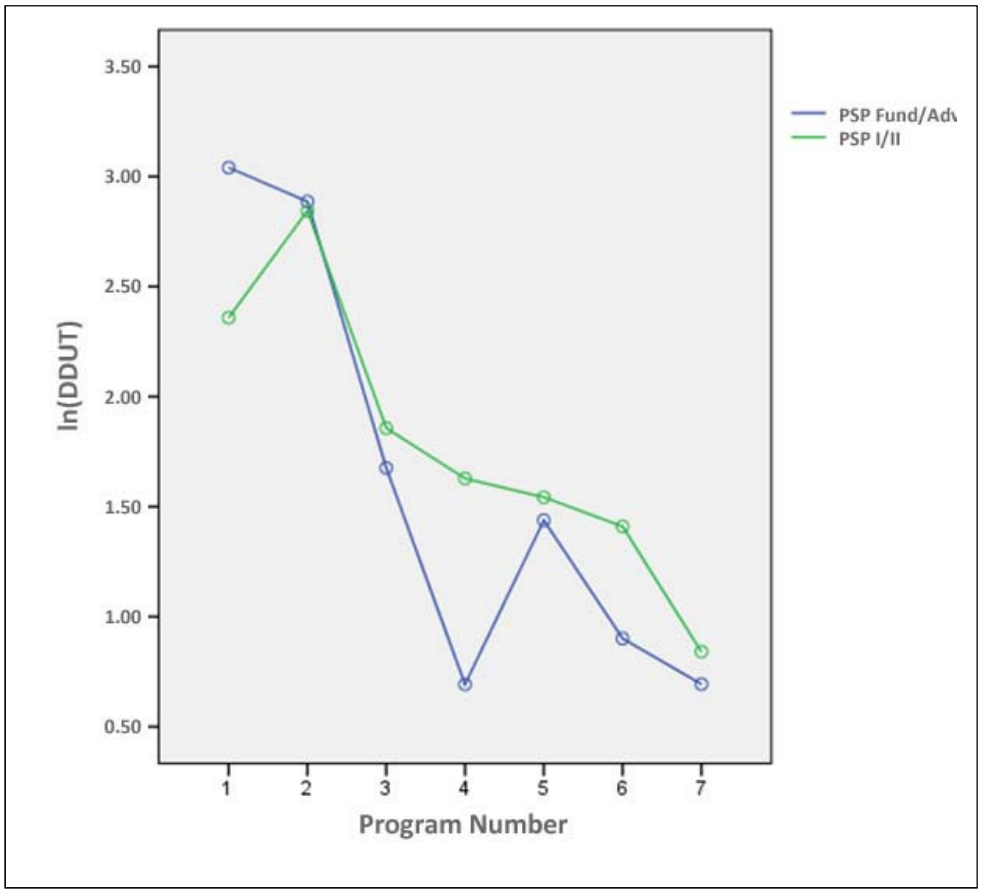

Figure 36: Comparison of Estimated Marginal Means of Ln(DDUT) versus Program Number between PSP Fund/Adv and PSP I/II

The third and last step of the analysis procedure consists of performing a two-way ANOVA grouping by PSP level and course to set bounds on the importance of PSP level as a predictor.

After this test, we found that there is significant difference between PSP0 and PSP1, between PSP0 and PSP2, and also between PSP0 and PSP2.1. Looking at it in more detail, results show that

- PSP1 was a factor of 0.2 more effective than PSP0 at an alpha level of 0.05 with a confidence range of the differences of [0.074, 0.939].
- PSP2 was a factor of 0.4 more effective than PSP0 at an alpha level of 0.05 with a confidence range of the differences of [1.028, 1.888].
- PSP2.1 was a factor of 0.45 more effective than PSP0 at an alpha level of 0.05 with a confidence range of the differences of [1.373, 2.133].

We also found that there is significant difference between PSP1 and PSP2, and between PSP1 and PSP2.1. Looking at it in more detail, results show that

- PSP2 was a factor of 0.22 more effective than PSP1 at an alpha level of 0.05 with a confidence range of the differences of [0.521, 1.381].
- PSP2.1 was a factor of 0.28 more effective than PSP1 at an alpha level of 0.05 with a confidence range of the differences of [0.866, 1.626].

Figure 37 shows the 95% confidence intervals of the log-transformation of defect density in unit testing for each PSP level, for both courses.

As a summary of this step, we can say that in both courses there is significant difference between all PSP levels, except between PSP2 and PSP2.1. The lack of significance between PSP2 and PSP2.1 may be because 1) the difference is too small to resolve (power of the sample), 2) the log-transformation hides the results (transforming those zeros will reduce the effect), or 3) there is no difference.

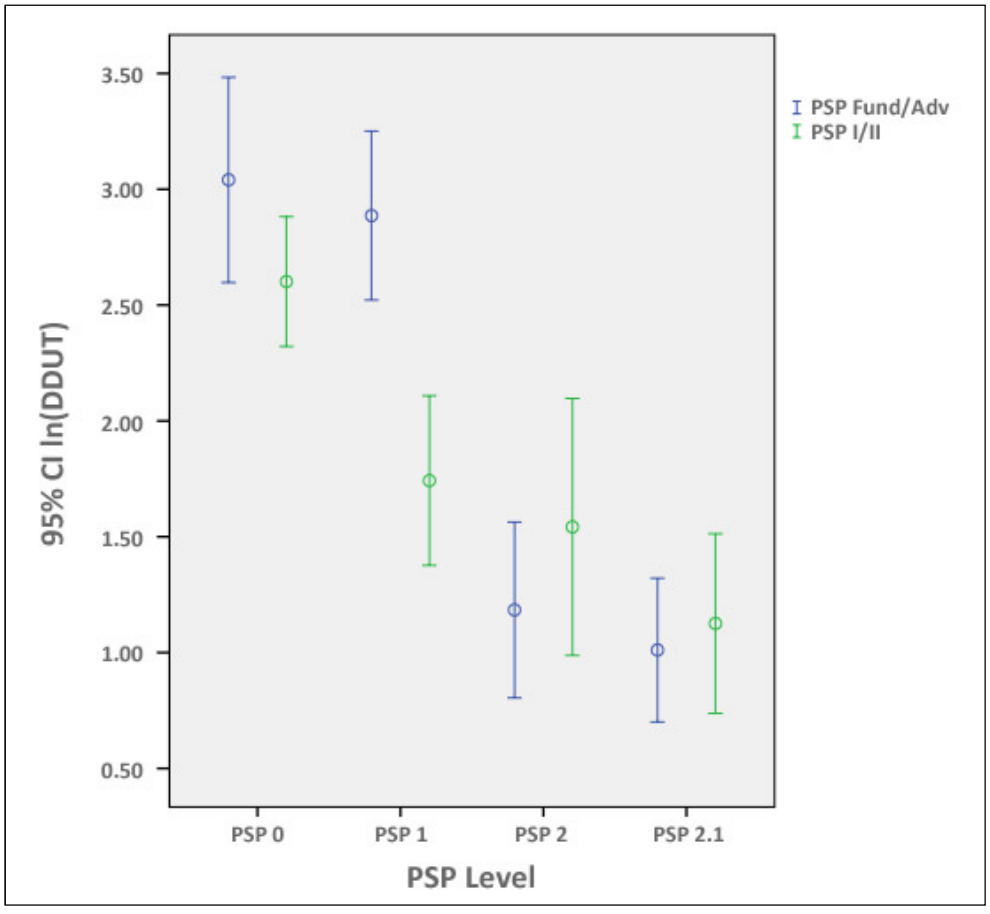

Figure 37: 95% Confidence Interval of Ln(DDUT) for each PSP Level in PSP Fund/Adv and PSP I/II

7.4 Threats to Validity and Limitations

By definition, defect density depends on the number of defects removed. But the number of defects found and removed in the test phase depends on the student's experience and on how good

the student is in doing unit testing. Therefore, we have a threat related to testing because the tests —those that are coincident with the treatment—may influence the student behavior.

According to the history, these courses were taught largely, but not entirely, at different times. Newer development environments and changes in the computer language instruction may alter subject behavior or the defect injection profile.

For a correct application of ANOVA, there is an assumption that the subjects are randomly selected for the treatments. We did not select the students; they were the ones that selected the course, and there is no precondition to do one course or another. So the random selection seems to be satisfied. But on the other hand, the students who took the PSP Advanced are more likely to go on to instruction or teaching. So, this group might respond better to the PSP instruction, and this could be seen as a threat to validity.

Even after transformation of the data, the normality assumption for ANOVA could not be satisfied. The distributions of defect density tend to be positively skewed, with long tails extending to the right and a truncated range at zero. This type of non-normal distribution is to be expected given the source of the data. There can never be a negative count for defects, so the truncation at zero is expected. In addition, we would expect many small values of defect density and relatively fewer large values. This positively skewed distribution is expected particularly when engineers (as a group) reduce the defect density of the programs as they improve their quality during the course. This is the type of data where either a logarithmic or inverse transformation can be used to create a more nearly normal distribution [Tabachnick 1989]. Based on our examination of the effects of these two types of transformations on the distribution of residuals, the logarithmic transformation was used in the confirmatory analysis.

We are comparing program assignments of two course versions as if they were identical. This can be considered a threat to validity. While the program assignments are very similar between the courses with the same programming exercise, they differ in the process elements and process used, so they are not exactly the same in both courses.

7.5 Conclusions

Previous studies of Personal Software Process [Hayes 1997, Rombach 2008] have examined the effect of the PSP on the performance of software engineers. The improvements, including product quality, were found to be statistically significant, and the observed results were considered generalizable beyond the involved participants. Those studies only considered students of the first version of the PSP course, which uses 10 program assignments and where there is a strong correlation between the program assignment and PSP level. Those studies may have some threats to external validity. In this work we try to face the generalization threat and consider the latest two course versions to see how different process introduction approaches correlate with quality results in PSP courses.

In this analysis we considered the work of 93 software engineers, who during PSP work developed eight or seven programs, depending on the course version. Each subject took the complete PSP course, either PSP for Engineers I and II or PSP Fundamentals and Advanced. We analyzed the data collected by each student to see how review and design improve the quality of a product in test.

Both courses appear to be effective in demonstrating use of design and reviews and both show reduction in defect injections. Levels achieved at end of the course are consistent with best-in-class practice. This cross-course comparison allowed us to discover that a "Hawthorne effect" is not as plausible as "gaining experience in the problem domain" or PSP techniques associated with PSP level as a causal explanation for the improvements. The strong association with PSP level suggests that learning effects are most plausible regarding mastering PSP techniques rather than general domain knowledge. This might be further examined in a future study with an analysis of phase injection and removal.

Because PSP level changes so rapidly in the PSP Fundamentals and PSP I, program number and PSP process level are tightly correlated in a way that makes separating the effects difficult. These results cannot ensure that the observed improvements are exclusively due to mastering the process techniques introduced in the PSP. We propose future analysis to obtain more generalizable results. The first approach would be a control experiment, consisting of introducing students to PSP, then remaining at PSP 1.0 through the first seven program assignments used in PSP I/II and PSP Fundamentals/Advanced. In this way, we can see how domain learning affects software quality improvement and then compare that result with the results of this study. A second approach would be an extended PSP course with at least three exercises at each PSP level. We believe that this would permit the process changes to stabilize so that we could more directly examine improvements between programs with and without process change.

In future analysis of these data from PSP Fundamentals/Advanced and PSP I/II, we will examine improvements in other dimensions, such as size estimation, effort estimation, defect yield, and productivity, to determine how effectively these courses work in those dimensions.

7.6 Acknowledgments

We thank Jim McCurley of SEI, Gabriela Mathieu, and Diego Vallespir of Universidad de la República for discussions of ANOVA analysis and suggestions of different statistical methods. We also thank the reviewers for their valuable contributions.

7.7 Author Biographies

Fernanda Grazioli
Graduate Student
Universidad de la República

Fernanda Grazioli is a graduate student at the Engineering School at the Universidad de la República, an honorary collaborator of the Software Engineering Research Group (GRIS), and a member of the Software and System Process Improvement Network in Uruguay (SPIN Uruguay).

William Nichols
Bill Nichols joined the Software Engineering Institute (SEI) in 2006 as a senior member of the technical staff and serves as a PSP instructor and TSP coach with the Team Software Process (TSP) Program. Prior to joining the SEI, Nichols led a software development team at the Bettis Laboratory near Pittsburgh, Pennsylvania, where he had been developing and maintaining nuclear engineering and scientific software for 14 years. His publication topics include the interaction patterns on software development teams, design and performance of a physics data acquisition

system, analysis and results from a particle physics experiment, and algorithm development for use in neutron diffusion programs. He has a doctorate in physics from Carnegie Mellon University.

[Batini 2006]
Batini, C. & Scannapieco, M. *Data Quality: Concepts, Methodologies and Techniques.* Springer-Verlag, 2006.

[Davis 2003]
Davis, B. N. & Mullaney, J. L. *The Team Software Process (TSP) in Practice: A Summary of Recent Results* (CMU/SEI-2003-TR-014), Software Engineering Institute, 2003. http://www.sei.cmu.edu/library/abstracts/reports/03tr014.cfm

[Hayes 1997]
Hayes, Will & Over, James. *The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers* (CMU/SEI-97-TR-001). Software Engineering Institute, Carnegie Mellon University, 1997. http://www.sei.cmu.edu/library/abstracts/reports/97tr001.cfm

[Humphrey 1995]
Humphrey, Watts S. *A Discipline for Software Engineering*. Addison-Wesley, 1995. http://www.sei.cmu.edu/library/abstracts/books/0201546108.cfm

[Kemerer 2009]
Kemerer, Chris & Paulk, Mark. "The Impact of Design and Code Reviews on Software Quality: An Empirical Study Based on PSP Data." *IEEE Transactions on Software Engineering 35*, 4 (July–August 2009): 534-550.

[Lee 2002]
Lee, Y. W.; Strong, D. M.; Kahn, B. K.; & Wang, R. Y. "AIMQ: A Methodology for Information Quality Assessment." *Information & Management, 40, 2* (December 2002): 133-146.

[Mayo 1949]
Mayo, E. *Hawthorne and the Western Electric Company*. "The Social Problems of an Industrial Civilization." Routledge, 1949.

[Neely 2005]
Neely, M. P. "The Product Approach to Data Quality and Fitness for Use: A Framework for Analysis," 221–236. *Proceedings of the 10th International Conference on Information Quality* Boston, MA, November 2005. MIT Press, 2005.

[O'Hara 2010]
O'Hara, R. B.; & Kotze, D. J. "Do not log-transform count data." *Methods in Ecology and Evolution 1* (2010): 118–122.

[Paulk 2010]
Paulk, Mark C. "The Impact of Process Discipline on Personal Software Quality and Productivity." *Software Quality Professional 12*, 2 (March 2010) 15–19.

[Paulk 2006]
Paulk, Mark C. “Factors Affecting Personal Software Quality.” *CrossTalk: The Journal of Defense Software Engineering 19*, 3 (March 2006): 9–13.

[Rombach 2008]
Rombach, Dieter; Munch, Jurgen; Ocampo, Alexis; Humphrey, Watts S.; & Burton, Dan. “Teaching Disciplined Software Development.” *The Journal of Systems and Software 81*, 5 (2008): 747–763.

[Strong 1997]
Strong, D. M.; Lee, Y. W.; & Wang, R. Y. “Data Quality in Context,” *Communications of the ACM 40*, 5 (1997): 103–110.

[Tabachnick 1989]
Tabachnick, B. G. & Fidell, L. S. *Using Multivariate Statistics.* Harper Collins, 1989.

[Wang 1995]
Wang, R. Y.; Reddy, M. P.; & Kon, H. B. “Toward Quality Data: An Attribute-Based Approach.” *Decision Support Systems13,* 3–4 (March 1995)*:*349-372.

[Wohlin 1998]
Wohlin, C. & Wesslen, A. “Understanding software defect detection in the Personal Software Process,” 49–58. *Proceedings of the Ninth International Symposium on Software Reliability Engineering*, Paderborn, Germany, November 1998. IEEE, 1998.