PSP Performance Analysis Report

Valentina Ivanova, PhD
New Bulgarian University

Contents:

Personal Software Process Overview

The Personal Software Process (PSP) is a structured software development process that is intended to help software engineers understand and improve their performance, by using a disciplined, data-driven procedure. The PSP was created by Watts Humphrey to apply the underlying principles of the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) to the software development practices of a single developer. It gives software engineers the process skills necessary to work on a Team Software Process (TSP) team. [1]

PSP training includes eight assignments in two courses – PSP Fundamentals and PSP Advanced. The report includes final analysis of all the data that was gathered during the training.

Planning Performance

Size Estimating Performance

The size of the implementation of the Assignments 2 to 7 (Table 1) is relatively small – between 74 and 182 added and modified LOC. Average size of the implementation of the Assignments is 109 LOC. Assignment 2, 3 and 5 are implemented by writing new code fragments. Assignments 4, 6 and 7 are implemented reusing existing code fragments.

	Plan		Size		Actual
	A&M	Estimated Size	LPI	UPI	A&M
Assignment 2	177.82				182
Assignment 3	110.95	108.4			96
Assignment 4	34.8	36.15			74
Assignment 5	87.19	67.75	22	151	93
Assignment 6	66.47	37.33	32	100	91
Assignment 7	104.9	81.01	68	141.7	118

Table 1

Size estimations for all Assignments from 5 to 7 were made using PROBE method A and are within the 70% statistical prediction intervals. Still the size estimates are under the actual size of the Added and Modified code. (Figure 1)

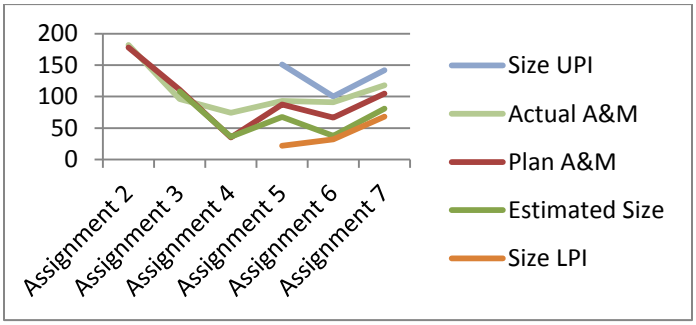

Figure 1

Size estimating error (Figure 2) tends to be +/-10% for Assignments that are implemented by writing new code fragments, while the size estimating error for Assignments that are implemented by reusing existing code fragments is times higher for some programs. The pick of +110% size estimating error is reached for Assignment 4 – when the concept of reusing code fragments was introduced. Assignment 6 reused code size estimation error is reduced but still high +40%. The final Assignment 7 is also implemented by reusing large segments of code. Size estimation error for this final assignment is close to the size estimation error of assignments implemented by writing new code (+ 10%).

My size estimating accuracy had evolved from totally inreliable for programs that reuse code fragments to a predictable accuracy of +/- 10% for both - programs based on new and on reused code. Still size estimation for programs that are reusing code fragments should be handled with special attention in order to keep the trend of reducing size estination error.

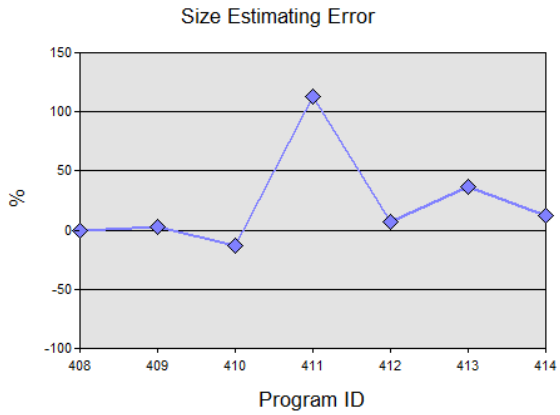

Figure 2

Time Estimating Performance

Time spend implementing Assignments 2 to 7 (Table 2) is between 101 and 279 minutes with average assignment implementation duration of 209 minutes.

	Plan	Estimated	Time		Actual
	Time	Time	LPI	UPI	Time
Assignment 2	330				279.37
Assignment 3	170.3	170.3			139.53
Assignment 4	52.44	52			101.82
Assignment 5	124.01	103	29	227	259
Assignment 6	125.53	120	4	246	210.45
Assignment 7	166.48	93	72.5	260	266.48

Table 2

Time estimation (Figure 3) for Assignment 3 and 4 are made using PROBE method C, time estimation for Assignment 5 is made using PROBE method A, and the last 2 Assignments (6 and 7) use PROBE method B for time estimation. Actual time of Assignment 5 exceeded the upper prediction interval because at that moment the process was changed. Design phase (including preparing the design templates) took 110 mutes, while the UPI was exceeded with about 30 minutes.

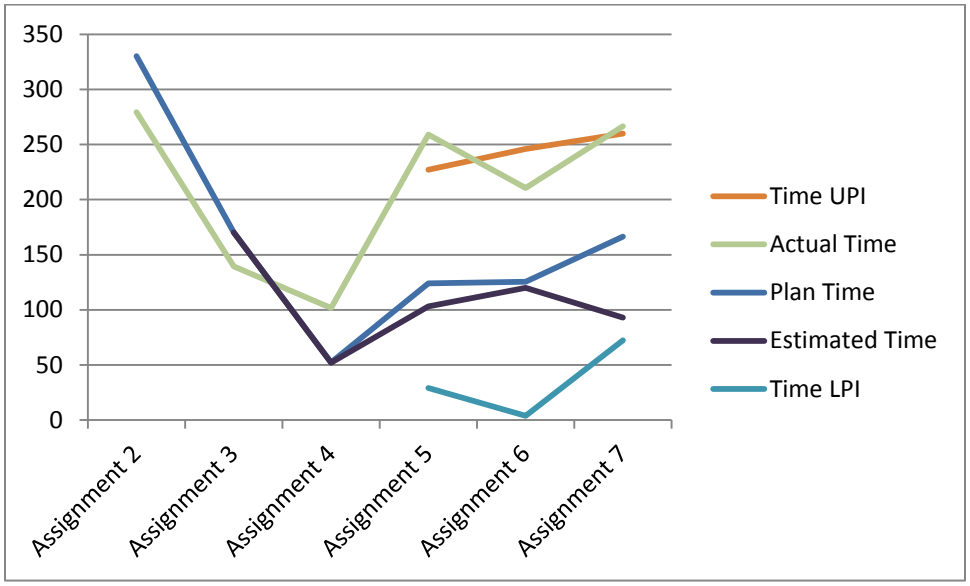

Figure 3

Time estimation error (Figure 4) varies from -18% (Assignment 3) up to + 108% (Assignment 5) due to constant changes of the process. After the final process change in Assignment 5, time estimation error decreases from + 108% for Assignment 5 to +60% for the final Assignment 7.

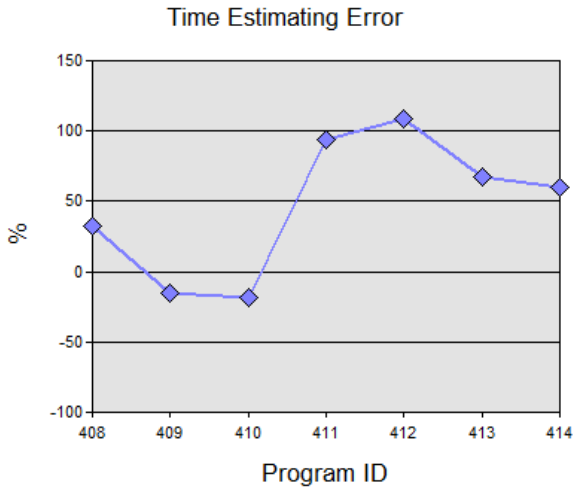

Figure 4

Following a constant process and size estimation with predictable accuracy will produce reliable data for accurate time estimation using PROBE methods.

Quality performance

Data about types of defects injected in each of the phases is stored in PSP Assignments_be.mdb file. The figures (Figure 5) show that the most common errors injected in DLD are function and interface errors.

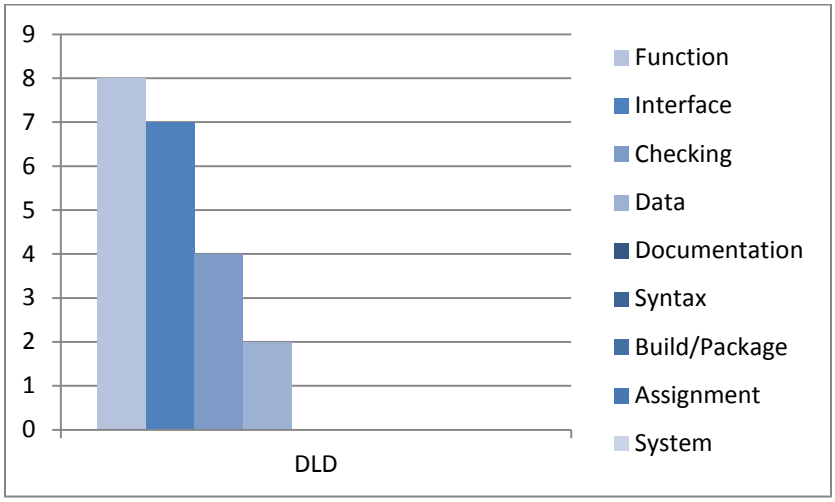

Figure 5

Most of the defects injected in code are functional (Figure 6). Significant number of defects injected is of type “assignments”, “checking” and “syntax”, followed by “interface” defects. In the early assignments “syntax” type was used to mark “user interaction input/output problem”, but later the same kind of defects were marked as “interface”, so cumulatively the “interface” defect type gets the highest number of injections during code phase.

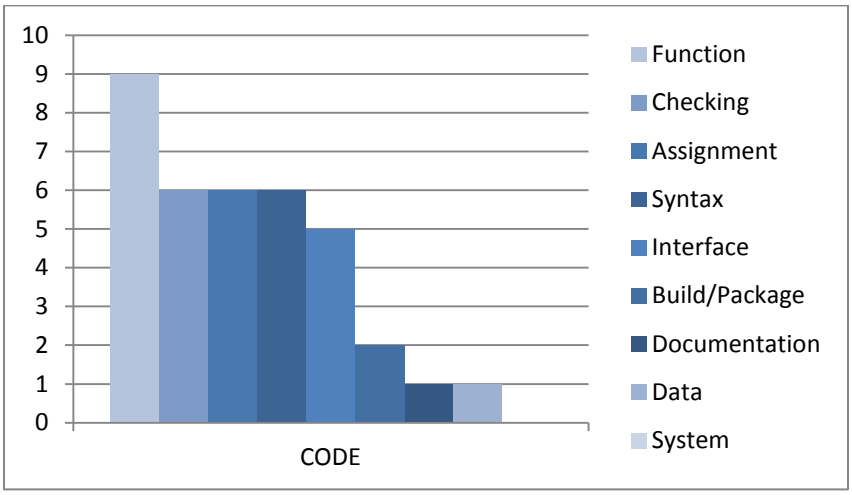

Figure 6

Defects per size unit (KLOC) or defect density distribution changed with the process changes (Table 3). Defect density in test decreased from pick levels of 66 defects per KLOC (Assignment 3) to 11 defects per KLOC (Assignment 6). The last assignments with constant process PSP 2.1 show that:

- Defects removed per KLOC during DLDR – 10-25
- Defects removed per KLOC during CR – 30-50 (twice the defects removed during DLDR)
- Defects removed per KLOC during test I – 10 -15

	Assignment 2	Assignment 3	Assignment 4	Assignment 5	Assignment 6	Assignment 7
DLDR		0	27	11	11	25.4
CR		62.5	0	43	33	51
TEST	26.8	65.9	20.8	13.5	10.75	16.9

Table 3

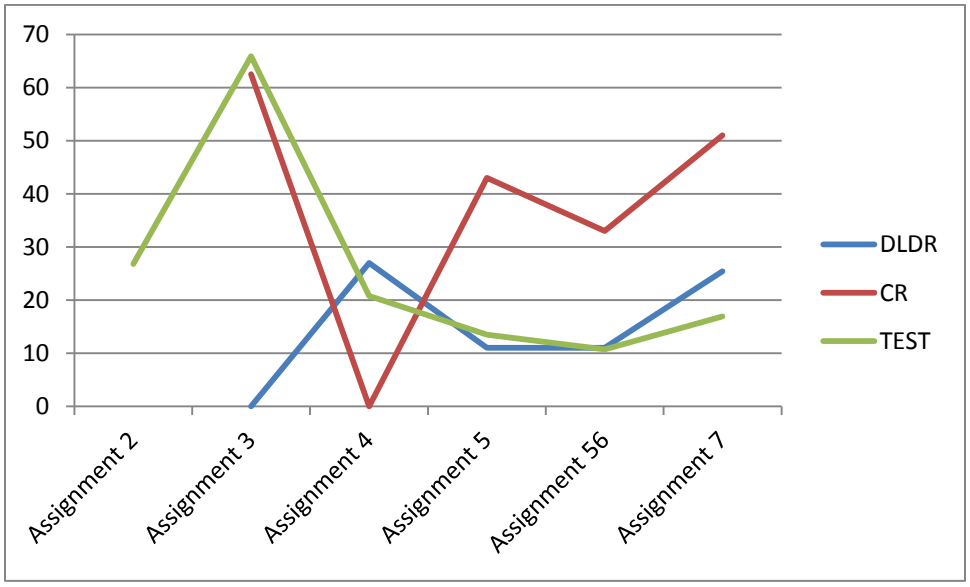

Figure 7

Defect density of the product (total defects per size unit) is relatively stable – about 75 defects per KLOC and vary from 40 (Assignment 4) to 102 (Assignment 7) defects per KLOC. Design reviews and code reviews are used for early removal of injected defects and reduce defect density in test.

Process performance

Defect removal rates per assignment (Table 4):

	DLDR defects/hr	CR defects/hr	UT defects/hr
Assignment 3	0	15	3.4
Assignment 4	7.5	0	4.1
Assignment 5	2.9	8.4	3.1
Assignment 6	2.8	10.2	2.4
Assignment 7	6.5	12.9	4.1

Table 4

To date defect-removal leverage for design reviews versus unit test is 1.2 and code review defect-removal leverage versus unit test is 2.4. In other words - during design reviews I'm 20% more effective in removing defects then in unit test. My data shows that in code reviews I'm two times and a half more effective in removing defects then in unit tests.

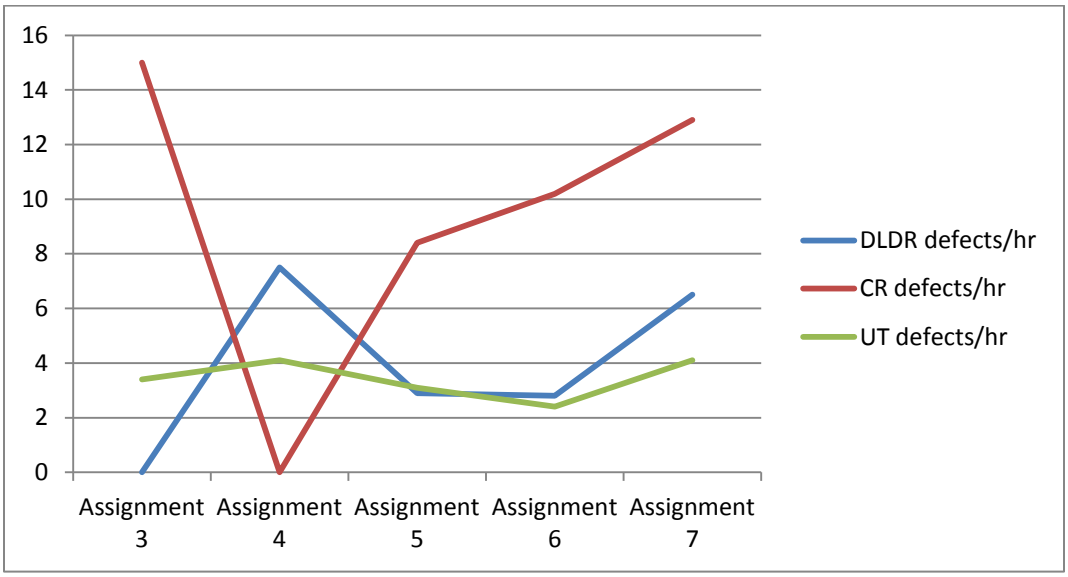

Figure 8

Code review and design review defect removal rates (Figure 8) tend to increase with my experience after a slight initial decrease in Assignment 5 - when the design templates were introduced.

Review rates for Assignment 5 to 7 are stable – around 250 LOC/ hour for design review and code review and about 125 LOC/hour for both (Figure 9).

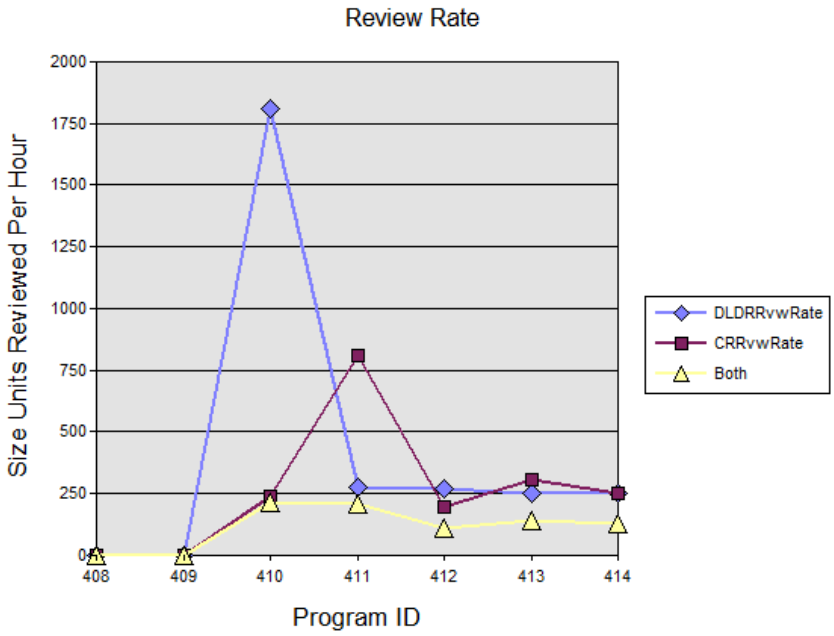

Figure 9

If the review rate is greater than 200 LOC/hour process yield is under 75%. If the rate is between 100 and 150 LOC/hour the yield is higher than 80% (Figure 10).

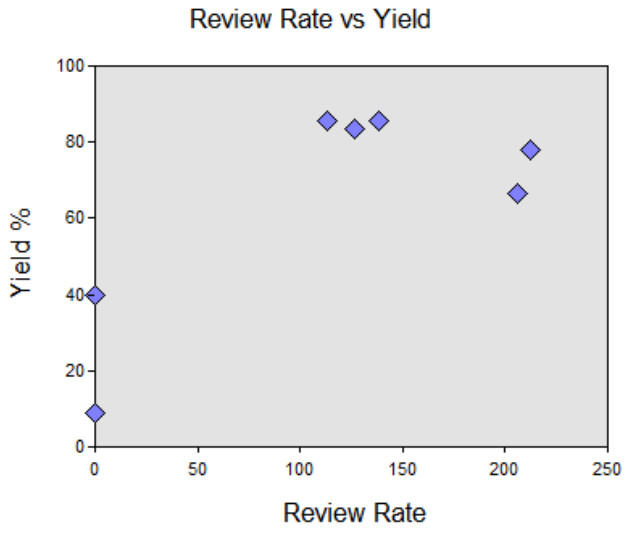

Figure 10

Appraisal to failure ratio (Table 5) shows the relation between time spend in design review and code review against the time spend in test (compile time is 0).

	A/F Ratio	Yield
Assignment 5	2.27	85.7
Assignment 6	1.53	85.7
Assignment 7	1.82	83.3

Table 5

In the last three Assignment that use PSP 2.1 process – time spend in appraisal was one time and a half to more than two times the time spend in test. The yield is relatively stable around 84%.

Process improvements

Performance improvement goals

1. Size estimation error under 5%
2. A/F Ratio above 2.5.
3. Process performance yield - 100%.

Process Improvement Proposals

1. Do a quick code review of the base code before planning the sizes of added and modified code fragments.
2. Add Design tests (DT) phase between DLD and DLDR.
3. Perform design checklists review and adjust the design checklist to PSP 2.1 detailed design process (incl. design templates).
4. Code review by module.
5. Perform cause-effect analysis for each defect found in test and add checklist items (in design review or code review checklists) for the causes.

Expected performance changes

Be able to do size and time estimations with 5% accuracy in order to support team planning process. Reach process yield of 97% and keep it.