

# Should we Teach the PSP<sup>SM\*</sup> for its Own Sake?

## –Position Paper–

Jürgen Börstler  
*Umeå University, Sweden*  
*jubo@cs.umu.se*

### 1. Introduction

The PSP (Personal Software Process) was developed as a means to improve the processes of individual software developers [1]. The PSP can be used as a tool to teach good software engineering practices leading to a predictable development process. Since 1994 PSP courses have been taught in industrial and academic environments. The results published so far are generally positive, but should still be taken with a grain of salt for several reasons.

- It is very difficult to validate PSP data points when data is collected manually. Results might therefore be based on error prone data [3].
- There are no “pure” PSP courses. It is therefore unclear which elements of a certain PSP course contributed to the positive results.
- There might be many negative experiences that have not been published (for whatever reasons).

### 2. Potential PSP problems

In principle we agree that the PSP is a valuable framework for the teaching of software engineering and software process improvement. The main message of the PSP, i.e. the value of a disciplined approach to software development, should be instilled in every future software engineer. Nevertheless, we have experienced several (real and potential) problems implementing the PSP.

- Students do not like it (see [4]). They perceive the PSP as tedious to use and complain that it takes away time from their “real work.” Especially for small exercises ( $\leq 100$  LOC) it is very difficult to convince students to use it.
- Data collection becomes very difficult in frequent edit-compile-debug cycles.
- Manual data collection becomes infeasible when class size exceeds 50 students.
- There is no room in typical computer science curricula for additional courses. A PSP course will therefore drive out other important courses.
- PSP requires basic programming knowledge, i.e. it is aimed at students who have a “flawed” process. Wouldn’t it be better to teach them a disciplined approach from the very beginning?

### 3. Approaches to teaching the PSP

A PSP course can be taught using quite different approaches.

- Focussing on the PSP itself, according to one of Watts Humphrey’s books ([1,2]).

- Focussing on software engineering principles in general and covering the PSP in accompanying exercises (as for example in [5]).
- Adapting the PSP exercises to cover an existing course.

At Umeå University we discussed how the basic principles of the PSP could be taught without interfering with our current computer science curriculum. We therefore experimented with a “downsized” PSP approach that could replace the exercises/lab part of an existing C++ course [4]. Our goals can be summarised as follows.

- Use an adaptation of PSP1 as the one and only PSP version.
  - One of the main lessons of the PSP is “plan and look what you did.” PSP1 with some minor modifications should be sufficient to deliver this message. Successive introduction of different PSP levels cannot be afforded, since it takes away time from the main contents of course.
- Cut down the number of exercises to 3-5.
  - When using one and the same PSP version throughout all exercises about four exercises should be sufficient collect historical data for performance evaluations.
- Provide a tool to support data collection to simplify collection and minimise data errors.
- Introduce the whole idea of using the PSP in one additional lecture (2x45 min).

Usage of the PSP was optional. Of the 78 students of the course, only six used the PSP throughout the course. Similar experiences were made in other courses where PSP usage was optional. The main reasons for abandoning the PSP were those listed in section2.

A more successful approach to familiarise students with the PSP was to develop tools to support the PSP in various project courses. This forced students to actively acquire information about the PSP and its usage.

We are now in the process of developing a new approach to teach the basic principles behind the PSP. The goal of this new approach is to teach a disciplined approach to software development from the very beginning. We have planned to redesign some PSP scripts and implement them as an electronic process guide that will guide the students through the practical part of a CS1 course.

## References

- [1] W.S: Humphrey: *A Discipline for Software Engineering*, Addison-Wesley, 1995.
- [2] W.S: Humphrey: *A Introduction to the Personal Software Process*, Addison-Wesley, 1997.
- [3] P.M. Johnson, A.M. Disney: A Critical Analysis of PSP Data Quality: Results from a Case Study, *Journal of Empirical Software Engineering* 4 (4), Dec 1999, 317-349.
- [4] S. Olofsson: Evaluation of the PSP in the Undergraduate Education, Technical Report UMNAD 272.99, Umeå University, Dept. of Computing Science, 1999.
- [5] C. Wohlin: The Personal Software Process as a Context for Empirical Studies, *IEEE TCSE Software Process Newsletter* 12, Spring 1998, 7-12.

---

\* PSP, Personal Software Process, TSPi, and Team Software Process are service marks of Carnegie Mellon University.