The Personal Software Process[1]

by
Watts S. Humphrey
watts@sei.cmu.edu
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

[1]This work is supported by the U.S. Department of Defense

Overview

The personal software process (PSP) has been developed by the Software Engineering Institute (SEI) to address the need for process improvement in small organizations and small project teams. This work started from the premise that improved personal process discipline can help to increase the effectiveness of individual engineers. It was also felt that as the individual engineers' performance improves, the performance of their teams and projects will also be more likely to improve.

The research work on the PSP started in 1989. It resulted in a method that is now being taught in several graduate university software engineering and computer science courses and experimentally tried at four corporations. While the PSP principles have been demonstrated with graduate software engineering students, their effectiveness has not yet been measured in industrial practice.

The current work on the PSP is focused on gaining experience with teaching the PSP and on identifying the most effective means for transitioning the approach into industrial practice. While the PSP methods are not complex and can be readily learned through self study, early indications are that a course environment is most effective. In particular, there is some evidence that many engineers will be more likely to personally adopt the methods if they have worked the PSP exercises in parallel with learning the methods. It is not yet clear what percentage of engineers will personally use the methods after completing the course but it appears that a reasonable number will. It is reasonably clear, however, that those engineers who do not work the exercises will not likely adopt the methods.

The Personal Software Process (PSP) Strategy

Today, when students start to program, they generally begin by learning a programming language. They practice on toy problems and develop the personal skills and techniques to deal with issues at this level. As they take more courses, they improve these methods and soon find they can develop fairly large programs relatively quickly. These skills, however, are inherently limited. While they may have sufficed on small-scale individual tasks, these programming-in-the-small skills do not provide an adequate foundation for the problems of large-scale multi-person projects. The PSP work follows a different strategy: it scales down industrial software practices to fit the needs of small-scale program development. By learning and using these practices on small programs, the students develop a foundation on which to build industrial-scale personal disciplines.

The Capability Maturity Model (CMM) provides a body of software engineering practices that have been found effective for large-scale software development. [Paulk] Starting with the CMM, the author selected and defined a subset of the CMM key process areas for use by individual software practitioners. [Humphrey 93, Humphrey 94] These methods have been structured in an evolving sequence of upward-compatible personal processes. Each process step is defined and used to guide engineers and students through a progressive series of process enhancements.

The professionals start by adjusting this simple pre-defined process to fit their current practices. This initial process includes basic measures of the time they spend in each PSP phase and the defects they find. They then gradually enhance this process through seven more PSP versions, adding a new software engineering method with each update. As they use these processes on ten software development exercises, they gather and analyze data on the work. Based on these analyses, they improve their processes for the next exercise. This provides the engineers with explicit feedback on the effectiveness of that process step. They also learn how to define and improve their personal processes.

Some CMM items are not included in the PSP because their effectiveness cannot be demonstrated at the individual level. Examples are subcontract management and

intergroup coordination. Others can be usefully practiced by individuals but their implications are better demonstrated in a small team environment. Requirements management and configuration management both fall in this category. While these are both critical topics, they are more effectively introduced after the initial PSP steps have been completed. Two other key process areas more directly relate to broader organizational issues. These are software quality assurance and training programs. While the PSP capabilities are directly relevant to these areas, it is not clear what useful exercises could be developed to demonstrate them at an individual level.

The Personal Software Process (PSP) Evolution

The PSP has a maturity structure much like the CMM. It is important to realize, however, that the PSP presumes an organization is at or near CMM level 2. Different numbers were selected for the PSP levels to avoid confusion with the CMM levels. The PSP progression is described in the following paragraphs.

PSP0 - The Baseline Process

The initial step in the PSP is to establish a baseline that includes some measurements and a reporting format. This provides a consistent basis for measuring progress and a defined foundation on which to improve. Following the first programming exercises, PSP0 is enhanced to PSP0.1 by adding a coding standard, size measurement, and the process improvement proposal (PIP).

PSP1 - The Personal Planning Process

PSP1 adds planning steps to PSP0. The initial step to PSP1 adds size and resource estimation. In PSP1.1, schedule planning and status tracking are also introduced.

PSP2 - Personal Quality Management

PSP2 adds personal design and code reviews to PSP1. These help the engineers to find defects earlier in their processes and to appreciate the benefits of finding defects early. They analyze the defects they found in the early programs and they use these data to establish review checklists that are tailored to their personal defect propensities.

The design process is addressed in PSP2.1. The intent is not to tell engineers how to do design but to address the criteria for design completion. That is, when they have finished the design, what must they have? In PSP2.1, design completeness criteria are established and various design verification and consistency techniques are examined.

PSP3 - A Cyclic Personal Process

To this point, the PSP has concentrated on a linear process for building small programs. In scaling the PSP2 up to larger projects, the approach is to subdivide larger programs into PSP2-sized pieces. These larger programs are then designed to be developed in incremental steps. This is done in PSP3, the cyclic development process. The first build is a base module or kernel that is enhanced in iterative cycles. In each iteration, a complete PSP2.1 is used, including design, code, compile, and test. Since each enhancement builds on the previously completed increments, the PSP3 process is suitable for programs of up to several thousand lines of code (KLOC). The cyclic PSP3 process can thus be an effective element of a large-scale development process.

Introduction Strategies

The PSP strategy is to start by introducing these methods in university curricula. This is initially being done at the graduate or senior undergraduate level. At a later time, the methods should be introduced at an earlier undergraduate level. This introduction, however, should be done in concert with basic programming methods and techniques.

Because it will likely take many years to introduce these methods through the educational system, the SEI is also supporting the transition of these methods into industrial practice and several corporations are currently participating in PSP introduction programs.

Status

The PSP work has been under development at the SEI since 1989 and the author has developed a total of about 25 KLOC of C++ and Object Pascal programs with the PSP method. Several graduate students at Carnegie Mellon University have participated in an introductory study of the PSP and early experimental work was started with Siemens Corporate Research and the AIS Corporation in Peoria, Ill. More recently, the Digital Equipment Corporation and Hewlett Packard Corporation have identified several interested projects and work is progressing on introducing the PSP in these organizations.

A textbook manuscript draft has been used to teach PSP courses at the University of Massachusetts and Howard University in the Fall of 1993 and at the Carnegie Mellon, Bradley, Embry Riddle, and McGill universities in the Spring of 1994. Based on the data from these courses, it is clear that both students and experienced engineers gain substantial benefits from this work. Quality improvements of two to three times are common and measured productivity improvements have averaged 35%. There are not yet sufficient data, however, to indicate the likely ranges of individual benefits or the factors that influence them. It also appears that the more experienced students make the greatest improvement. In one case, an experienced engineer found that by using the PSP, he reduced his numbers of test defects by about ten times and more than doubled his productivity.

Conclusions

The early work on the PSP indicates that a structured, disciplined, and measured personal software process can provide the guidance and feedback needed to help engineers improve their personal performance.

- The engineers are more aware of their work and better able to understand areas where they can improve.
- The process measures provide them direct and explicit feedback on their work.
- This rapid feedback reinforces the use of sound engineering practices.
- A course format has also been found to be the most effective for introducing the PSP in industrial organizations.
- Several engineers have attempted to use PSP methods in their regular work. Without the rapid feedback provided by the PSP exercises, however, they have generally been unable to sustain the needed process discipline.

The PSP is not a magic answer to the problems of developing good software. The methods take time and effort to learn and they require consistent discipline to use. While the initial PSP work concentrates on the design, code, and test phases of software development, the PSP principles can be applied to requirements specification, product maintenance, test planning, documentation development, or many other aspects of the software process. The detailed design, code, and unit test phases were selected because they are important development phases and they are most suitable for the small but challenging classroom exercises.

References

[Humphrey 93] W. S. Humphrey, "The Personal Software Process, Rationale and Status," The 8th International Software Process Workshop, Wadern, Germany, March 2-5, 1993.

[Humphrey 94] W. S. Humphrey, "Process Feedback and Learning," The 9th International Software Process Workshop, Airlie, VA, Oct. 5-7, 1994.

[Paulk] M. C. Paulk, Bill Curtis, M. B. Chrisis, "Capability Maturity Model for Software, Version 1.1," Software Engineering Institute Technical Report, CMU/SEI-93-TR-24, February 1993.

 Software Process Newsletter, Technical Council on Software Engineering, IEEE Computer Society, Volume 13, No. 1, Sept. 1994, pp SPN 1-3.