Personal software process

The **Personal Software Process (PSP)** is a structured software development process that is designed to help software engineers better understand and improve their performance by bringing discipline to the way they develop software and tracking their predicted and actual development of the code. It clearly shows developers how to manage the quality of their products, how to make a sound plan, and how to make commitments. It also offers them the data to justify their plans. They can evaluate their work and suggest improvement direction by analyzing and reviewing development time, defects, and size data. 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 claims to give software engineers the process skills necessary to work on a team software process (TSP) team.

"Personal Software Process" and "PSP" are registered service marks of the Carnegie Mellon University.[1][2]

## Contents

- **Objectives**
- **PSP structure**
- **The importance of data**
- **Planning and tracking**
- **Using the PSP**
  - PSP and the TSP
  - PSP and other methodologies
  - Quality
- **Certification**
- **See also**
- **References**
- **Further reading**
- **External links**

### Objectives

The PSP aims to provide software engineers with disciplined methods for improving personal software development processes. The PSP helps software engineers to:

- Improve their estimating and planning skills.
- Make commitments they can keep.
- Manage the quality of their projects.
- Reduce the number of defects in their work.
PSP training follows an evolutionary improvement approach: an engineer learning to integrate the PSP into his or her process begins at the first level – PSP0 – and progresses in process maturity to the final level – PSP2.1. Each Level has detailed scripts, checklists and templates to guide the engineer through required steps and helps the engineer improve their own personal software process. Humphrey encourages proficient engineers to customize these scripts and templates as they gain an understanding of their own strengths and weaknesses.

**Process**

The input to PSP is the requirements; requirements document is completed and delivered to the engineer.

**PSP0, PSP0.1 (Introduces process discipline and measurement)**

PSP0 has 3 phases: planning, development (design, code, compile, test) and a post mortem. A baseline is established of current process measuring: time spent on programming, faults injected/removed, size of a program. In a post mortem, the engineer ensures all data for the projects has been properly recorded and analysed. PSP0.1 advances the process by adding a coding standard, a size measurement and the development of a personal process improvement plan (PIP). In the PIP, the engineer records ideas for improving his own process.

**PSP1, PSP1.1 (Introduces estimating and planning)**

Based upon the baseline data collected in PSP0 and PSP0.1, the engineer estimates how large a new program will be and prepares a test report (PSP1). Accumulated data from previous projects is used to estimate the total time. Each new project will record the actual time spent. This information is used for task and schedule planning and estimation (PSP1.1).

**PSP2, PSP2.1 (Introduces quality management and design)**

PSP2 adds two new phases: design review and code review. Defect prevention and removal of them are the focus at the PSP2. Engineers learn to evaluate and improve their process by measuring how long tasks take and the number of defects they inject and remove in each phase of development. Engineers construct and use checklists for design and code reviews. PSP2.1 introduces design specification and analysis techniques

(PSP3 is a legacy level that has been superseded by TSP.)

**The importance of data**

One of the core aspects of the PSP is using historical data to analyze and improve process performance. PSP data collection is supported by four main elements:

- Scripts
- Measures
- Standards
- Forms

The PSP scripts provide expert-level guidance to following the process steps and they provide a framework for applying the PSP measures. The PSP has four core measures:
- Size – the size measure for a product part, such as lines of code (LOC).
- Effort – the time required to complete a task, usually recorded in minutes.
- Quality – the number of defects in the product.
- Schedule – a measure of project progression, tracked against planned and actual completion dates.

Applying standards to the process can ensure the data is precise and consistent. Data is logged in forms, normally using a PSP software tool. The SEI has developed a PSP tool and there are also open source options available, such as Process Dashboard.

The key data collected in the PSP tool are time, defect, and size data – the time spent in each phase; when and where defects were injected, found, and fixed; and the size of the product parts. Software developers use many other measures that are derived from these three basic measures to understand and improve their performance. Derived measures include:

- estimation accuracy (size/time)
- prediction intervals (size/time)
- time in phase distribution
- defect injection distribution
- defect removal distribution
- productivity
- reuse percentage
- cost performance index
- planned value
- earned value
- predicted earned value
- defect density
- defect density by phase
- defect removal rate by phase
- defect removal leverage
- review rates
- process yield
- phase yield
- failure cost of quality (COQ)
- appraisal COQ
- appraisal/failure COQ ratio

**Planning and tracking**

Logging time, defect, and size data is an essential part of planning and tracking PSP projects, as historical data is used to improve estimating accuracy.

The PSP uses the PROxy-Based Estimation (PROBE) method to improve a developer's estimating skills for more accurate project planning. For project tracking, the PSP uses the earned value method.

The PSP also uses statistical techniques, such as correlation, linear regression, and standard deviation, to translate data into useful information for improving estimating, planning and quality. These statistical formulas are calculated by the PSP tool.
Using the PSP

The PSP is intended to help a developer improve their personal process; therefore PSP developers are expected to continue adapting the process to ensure it meets their personal needs.

PSP and the TSP

In practice, PSP skills are used in a TSP team environment. TSP teams consist of PSP-trained developers who volunteer for areas of project responsibility, so the project is managed by the team itself. Using personal data gathered using their PSP skills; the team makes the plans, the estimates, and controls the quality.

Using PSP process methods can help TSP teams to meet their schedule commitments and produce high quality software. For example, according to research by Watts Humphrey, a third of all software projects fail,[3] but an SEI study on 20 TSP projects in 13 different organizations found that TSP teams missed their target schedules by an average of only six percent.[4]

Successfully meeting schedule commitments can be attributed to using historical data to make more accurate estimates, so projects are based on realistic plans – and by using PSP quality methods, they produce low-defect software, which reduces time spent on removing defects in later phases, such as integration and acceptance testing.

PSP and other methodologies

The PSP is a personal process that can be adapted to suit the needs of the individual developer. It is not specific to any programming or design methodology; therefore it can be used with different methodologies, including Agile software development.

Software engineering methods can be considered to vary from predictive through adaptive. The PSP is a predictive methodology, and Agile is considered adaptive, but despite their differences, the TSP/PSP and Agile share several concepts and approaches – particularly in regard to team organization. They both enable the team to:

- Define their goals and standards.
- Estimate and schedule the work.
- Determine realistic and attainable schedules.
- Make plans and process improvements.

Both Agile and the TSP/PSP share the idea of team members taking responsibility for their own work and working together to agree on a realistic plan, creating an environment of trust and accountability. However, the TSP/PSP differs from Agile in its emphasis on documenting the process and its use of data for predicting and defining project schedules.

Quality

High-quality software is the goal of the PSP, and quality is measured in terms of defects. For the PSP, a quality process should produce low-defect software that meets the user needs.
The PSP phase structure enables PSP developers to catch defects early. By catching defects early, the PSP can reduce the amount of time spent in later phases, such as Test.

The PSP theory is that it is more economical and effective to remove defects as close as possible to where and when they were injected, so software engineers are encouraged to conduct personal reviews for each phase of development. Therefore, the PSP phase structure includes two review phases:

- Design Review
- Code Review

To do an effective review, you need to follow a structured review process. The PSP recommends using checklists to help developers to consistently follow an orderly procedure.

The PSP follows the premise that when people make mistakes, their errors are usually predictable, so PSP developers can personalize their checklists to target their own common errors. Software engineers are also expected to complete process improvement proposals, to identify areas of weakness in their current performance that they should target for improvement. Historical project data, which exposes where time is spent and defects introduced, help developers to identify areas to improve.

PSP developers are also expected to conduct personal reviews before their work undergoes a peer or team review.

### Certification

A certification covering PSP is offered by the SEI at Carnegie Mellon University. The steps to becoming an SEI-Certified PSP Developer are: learn the PSP; take the certification exam; maintain credentials. The PSP Developer examination is based on concepts found in the PSP Body of Knowledge.[5] The SEI maintains an FAQ[1] on certification.

### See also

- Agile software development
- Capability Maturity Model Integration (CMMI)
- Carnegie Mellon University
- Proxy-based estimating (PROBE)
- Software Engineering Institute (SEI)
- Team software process (TSP)
- Watts Humphrey

### References

Further reading

- "Personal Quality Management with the Personal Software Process" (http://www.methodsanalystools.com/archive/archive.php?id=60) article.

External links

- Software Process Dashboard (http://www.processdash.com/). Open-source (GPL3) PSP and TSP tool; offered both without and with proprietary SEI scripts, latter requiring free SEI registration.