

Personal Software Process

A 10-Week Program

Classes Meet on Fridays

September 11 thru
November 20, 1998

UT Pickle Research Campus
Austin, Texas

sponsored by
Software Quality Institute
THE UNIVERSITY OF TEXAS AT AUSTIN

A New, Practical Technology for Software
Practitioners

Developed by the Software Engineering Institute (SEI)

Brings discipline to individual software engineers

Dramatically improves the quality, predictability and cycle
time of software-intensive systems

Increases awareness of processes used and improves the
performance of those processes

PSP Helps Developers Do a Better Job!

Learn how to set personal goals for improvement

How to measure your work

Adjust your process to meet your goals

A Strategy for Engineers' Self-Development and Enhanced Productivity

The Personal Software Process (PSP) is a defined and measured software process designed to be used by individual software practitioners. Its intended use is to guide the planning and development of software modules or small programs.

With PSP, practitioners develop software using a disciplined, structured approach. They follow a defined process, plan, measure, and track their work, manage product quality, and apply quantitative feedback to improve their personal work processes.

The goal of the PSP is to provide individual software practitioners with a method that allows them to predictably and continuously improve the quality of their work and their ability to accurately estimate and achieve schedules.

PSP - A New Technology

The Personal Software Process (PSP) is a new technology developed by the Software Engineering Institute (SEI) that brings discipline to individual software engineers, dramatically improving the quality, predictability, and cycle time of software-intensive systems.

PSP Increases Accuracy and Productivity

The Personal Software Process (PSP) is a structured set of forms, standards, and procedures that is designed to help you do better work.

Data on the PSP's early use show that engineers achieve average reductions of 75% in numbers of injected defects, make more accurate plans, and have higher productivity. To date, the PSP has been used to write more than 1000 small programs with a total of over 100,000 LOC (lines of code).

PSP Converts Craft To Structured Discipline

The current practice of software engineering is more of a craft than a structured engineering discipline. Personal Software Process training provides a step-by-step framework that demonstrates the methods of disciplined software engineering. Participants learn by using their own data how effective the methods can be for them and their organizations.

Personal Software Process (PSP) and quality management methods are described in *A Discipline for Software Engineering* by Watts Humphrey. The PSP is a scaled-down version of industrial software process that is based on quality management principles and is designed to be used by an individual software engineer.

Using the discipline of the PSP, engineers learn to continuously improve the quality and reliability of their own processes. A tenfold reduction in test defects and an average 25% increase in productivity are the norm for PSP students. Organizations that train their software engineers to use the PSP can

expect substantial improvements in product quality, cost performance, and schedule performance. Test and rework efforts will also be reduced resulting in improved cycle time.

PSP Training: Hands-On Practical, Learning by Doing

Jim Terrel, PSP trainer for the Software Quality Institute, explains:

The PSP training is really very much a hands-on, learn-by-doing experience.

Each student writes 10 programs during the course. The first program is used to baseline the individual's current process. Then the student is incrementally introduced to new process steps to capture data required for improvement.

Several in-depth analyses and reports concerning the student's personal data are also required. These tie the data together and show the person where and how to improve his or her process.

There are actually two major thrusts to PSP: improving planning and improving quality. The underlying notion is that the quality of a software system is governed by the quality of its worst components -- the old weakest-link-in-the-chain idea.

PSP training is structure around the dual planning and quality goals, with planning emphasized in the first half of the course and quality in the second half.

It's like this. The quality of each component is governed by the number of defects injected and removed by the individual who constructed the component, and the rates of defect injection and removal are affected by the individual's knowledge, discipline, and commitment. If the person doesn't know how to do something -- that is, lacks the knowledge - then he or she needs training if this individual is to create high-quality products.

This really isn't a PSP issue except in the sense that a person may have the computer science knowledge but lack the programming skills to produce high-quality products. By creating a stable personal software development process that can be analyzed, a person can better apply his or her knowledge.

If the individual lacks discipline, then he or she tends to make lots of sloppy errors. The person knows better but hasn't taken the time or applied the effort to find the defects early in his or her own process. "The compiler or unit test will find 'em."

Likewise, if the person has accepted an unreasonable commitment due to schedule pressure or just poor scoping and estimation of the time and effort involved to perform a task, the individual is also likely to produce low-quality products in an attempt to meet the commitment made to the manager.

Thus, by standardizing the activities I perform and collecting data (time, defect types, etc.), I am creating a repeatable personal process that I can analyze and so improve my planning and quality.

PSP Training Topics

Planning

Introduction to PSP

Software size measurement

Estimating software size

Resource and schedule planning

Measurements

Quality

Code reviews

Quality management

Design notation

Design framework

Detailed design reviews

Cyclic and team processes

Process development

Using the PSP in an organization

SQI Offers 10-Week PSP Training in Austin

The Software Quality Institute (SQI) at The University of Texas at Austin is licensed to offer PSP training all-day on Fridays for 10 weeks: September 11–November 20, 1998. The classes will meet at UT's Pickle Research Campus in northwest Austin. Hours are 8:30 a.m.–5:00 p.m. with lunch included. Instructor is Jim Terrel of Cedar Creek Software. Jim is a software practitioner of many years' experience who received training at the Software Engineering Institute (SEI) as an authorized PSP trainer.

Intended Audience

Programmers, software engineers, and software designers are the target audience for the PSP for engineers training. Participants will be required to do a significant amount of coding to successfully complete the course.

Practical, Hands-On Workshop Training

Students will want to use in the PSP class the same language they are currently using in the workplace in order to facilitate the transfer of learning from the classroom.

C is the language most commonly used but C++, Ada, Pascal, FORTRAN, and Visual Basic have also been used.

Laptops, Software, and Homework Assignments

Participants must have a portable computer with Windows environment and an installed copy of Microsoft Excel. Students will use in class the same programming language they normally use in the workplace.

SQI will send a copy the textbook *A Discipline for Software Engineering* by Watts Humphrey to participants upon registration and about eight hours of readings will be required before the class begins.

Participants will be assigned homework each week after each class, due by 5:00 p.m. on the following Monday. Weekly homework assignments will be graded and students should plan to set aside an estimated eight hours over the weekend to complete the assignments.

Certificates Awarded

Students who successfully complete the PSP training will be awarded certificates.

The Personal Software Process Overview, Practice, and Results

by Watts S. Humphrey, Author of *A Discipline for Software Engineering*

You would probably agree if I told you that the number of defects to be found in testing a program would be proportional to the number in the product when it entered test. It seems reasonable to find more defects when there are more to be found. If, however, I said that the number of defects in the product after test would be proportional to the number on test entry, many software engineers would not agree.

Software people act as if testing will find all or most of a product's defects. There is, however, compelling evidence that even well-run unit tests are less than 70 percent effective at finding defects. Integration and system tests only find about 45 percent and function tests typically find a dismal 8 percent of the product's defects. Thus, to get a quality product out of test you must put one in. If software engineers really believed this, they would act more like quality engineers in other fields. That is, they would concentrate on finding or preventing the defects before the start of test.

Many software engineers will argue that this approach is impractical for software development. There is, however, evidence that this strategy works. By using a defined and measured personal software process, engineers can improve the quality of their products by five to ten times while also improving their productivity. This personal software process (PSP) is a promising way for engineers to understand their own performance and to see how to improve it. The PSP is new, however, and there is limited experience

with its introduction and use. It has been taught in six universities and experimentally applied by three software organizations.

The original impetus for developing the PSP came from questions about the Software Engineering Institute's (SEI) capability maturity model (CMM). Many viewed the CMM as designed for large organizations and did not see how it could be applied to individual work or to small project teams. While the CMM does apply to both large and small organizations, more explicit guidance was clearly needed. The SEI thus started a process research project to examine ways individual engineers could apply level 5 process principles. After several years of research, means were devised to adapt 12 of the 18 CMM key process areas to the work of individual software engineers.

Experimental work was then started with several corporations to see how experienced engineers would react to the PSP and to explore introduction methods. It was found that experienced engineers are generally attracted by the PSP strategy and find the methods help them in their work. In the words of one engineer, "This isn't for the company, it's for me."

The PSP applies process principles to the work of software engineers by providing a defined personal process framework, introducing a family of process measures, using these measures to track and evaluate performance, striving to meet quality criteria and improvement goals. In using the PSP, engineers develop a plan for every project, record their development time, retain the data in project summary reports, use the data to plan future projects, analyze the data to evolve their processes and improve their performance. In the PSP, project plans include a documented size estimate and a statistically derived size prediction interval. Historical data are used to estimate the development time and to calculate the time prediction interval. Based on the engineer's personal data on prior projects, these time data are spread over the project phases. Together with data on the engineer's prior commitments and available working time, the engineer then produces a schedule.

The development plan also includes a defect estimate. From data on their prior experience, the engineers learn to accurately project the defects they will inject and remove per phase. They also know the likely distribution of defect types and the likely times required to find and fix defects in review, compile, and test. As they track these data, the engineers develop review checklists to help find the defects earlier in their processes. They also look for ways to improve their processes so they can prevent the defects before they are introduced.

The quality strategy used with the PSP is consistent with that practiced in many hardware organizations: build quality into the product from the start. The general practice in software has been to design and implement products as rapidly as possible and then to rely on compile and test to find the defects. When organizations work this way they spend as much as half their development resources compiling and testing. Even after all this expense, quality is generally still so poor that extensive field tests are needed before products can be offered for general use. PSP engineers follow a different strategy. They have found that testing is inefficient and marginally effective. Their objective is to remove all defects before the first compile or test. One of the principal PSP quality measures is yield: the percent of all defects removed before the first compile or test.

Registration Information

The registration fee covers the cost of the textbook, course materials, and light refreshments. Early registration is encouraged; enrollment is limited. Registration includes completed form, payment by check or credit card (with expiration date), purchase order, request for invoice or approved government voucher or UT IDT information.

To register: Mail registration form and payment information to address below or fax to (512) 471-4824.

Registration is not complete until we receive payment. We will acknowledge your registration with a confirmation letter. Payment receipts will be mailed upon request. If you do not receive a confirmation letter, call (512) 475-6779 to confirm your registration. Persons requiring special accommodations should notify us in writing two weeks prior to the start of the program. Registration deadline: September 7, 1998.

Refund Policy: Registrants who cancel will receive a refund less a \$75 processing fee if we receive written notification postmarked or delivered on or before September 7, 1998. No refund will be granted after September 7, 1998. If the class is canceled, a full refund will be given. Substitutions may be made at any time. The Software Quality Institute has the right to substitute instructors, change the date the course meets, and cancel courses due to insufficient enrollment or unforeseen events.

For additional information or questions, please contact:

Software Quality Institute
The University of Texas at Austin
PRC MER Mail Code R9800
Austin, TX 78712-1080
Telephone: (512) 475-8649 or (800) 687-8012
Fax: (512) 471-4824
E-mail: elms@sqi.utexas.edu
Web: <http://www.utexas.edu/coe/sqi>
Federal Tax ID #74-6000203
State of TX Vendor ID #3721 721 721 7482

Personal Software Process Registration

The University of Texas at Austin, Software Quality Institute *Please call to reserve a seat in advance: (512) 475-8649*

The registration fee is \$3,000 for nongovernment; \$2,700 for government— 20% discount for 5 or more from same organization.

Name _____	Title _____
Organization _____	Billing address _____
Business address _____	_____
_____	Attention _____
Business phone _____	Fax _____
Home phone _____	e-mail _____