The Personal Software Process (PSP) Tutorial

Watts Humphrey / Jim Over
Speaker: Daniel M. Roy (STPP, visiting scientist SEI)

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Sponsored by the U.S. Department of Defense
Tutorial Objectives

To describe the personal software process (PSP)

To show where and how the PSP can be used to improve individual software engineering performance

To show how the PSP can enable software organizations to improve their capability

To describe the status and plans for the PSP
Tutorial Agenda

PSP Description
• PSP overview
• PSP processes
• PSP planning and quality management

PSP Introduction
• course and industry data
• introducing PSP
• PSP and process improvement
The Software Problem -1

All businesses are becoming software businesses.

The business importance of software is increasing.
• software is now the critical element in many products
• software cycle time often exceeds hardware cycle time
The Software Problem - 2

Software quality limits our ability to field many critical systems.

Poor software quality in fielded systems is expensive.

- reduced customer satisfaction
- delayed shipments
- expensive service and enhancement
- potential for accident or loss of life
The Software Problem - 3

It is now generally recognized that an effective way to improve the software business is by improving the development and maintenance processes.

Organizational progress with process improvement is limited because
- there is limited process improvement experience
- process improvement takes time
- process improvement efforts are hard to sell
The Software Problem - 4

The PSP addresses these problems by

• providing convincing evidence of the benefits of process improvement
• exposing the engineers to the benefits of using effective processes in their work
• teaching the engineers effective process improvement methods
• providing the historical data to better manage cost, schedule, and quality
The PSP Paradigm

The PSP is based on process improvement principles.

- practitioners establish personal process goals
- they define the methods they will use
- they measure their work
- they analyze the results
- based on these analyses, they adjust their methods to better meet their personal goals
The PSP Strategy

Start with the engineer’s current process

Gradually introduce new methods

Practice these methods on module-sized programs

The engineers then see for themselves how these methods help them.
PSP Overview - 1

The PSP is a process for individuals to use.

It applies to most structured personal tasks.
• writing small programs or documents
• defining requirements or processes
• conducting reviews or tests, etc.

It is extendible to team development of large-scale software systems.

It is a SW CMM Level 5 process for individual engineers.
PSP Overview - 2

Individuals learn the PSP in 7 process steps.

They write 10 module-sized programs using these PSP steps.
• they gather and analyze data on their work
• based on these analyses they improve their working methods

The PSP exercises provide the rapid feedback needed for effective learning.
The PSP Course - 1

The PSP is best introduced with a course format.

Prerequisites
• fluency with one programming language
• basic understanding of programming design
• mathematics through integral calculus

Additional background is helpful.
• statistics
• project management
• formal methods
The PSP Course - 2

Facilities required
• personal computing capability
• spreadsheet and database support
• development environment

Academic course
• one semester workload
• 15+ 90 minute lectures
• laboratory time of 5+ hours per week
• study time of 5+ hours per week
The PSP Course - 3

Industry course formats
• two one-week courses with 20+ hours of post course homework per week
• no homework format consisting of four sessions “3-4-3-4”
• one day per week format
• two day per week format
The PSP Is an Evolving Process

- PSP0: Current process
  - Basic measures

- PSP1: Task planning
  - Size estimating
  - Test report
  - Schedule planning

- PSP1.1: Design templates
  - Design reviews

- PSP2: Code reviews
  - Cyclic development
  - Design templates

- PSP3: Team Software Process
  - Process improvement proposal
  - Size measurement
  - Coding standard

© 1998 Carnegie Mellon University
The PSP0 Process

With PSP0, engineers use their current design and development methods.

They gather data on their work.
- the time spent by phase
- the defects found in compile and test

They analyze and report these data.
The PSP0 Process Lessons

With PSP0, engineers learn to use a basic personal process.

• they gather data on their personal work
• they learn how and why to measure the sizes of the products they produce
• they gather baseline data on their personal processes
The PSP1 Process

The PSP0 is augmented to include
- coding standards
- size estimating
- resource estimating
- schedule estimating
- earned value tracking
- process improvement proposal (PIP)
- test report
The PSP1 Lessons

With PSP1, engineers estimate the sizes and development times of the products they produce.
  • they use their historical data to improve their estimates
  • they project the likely statistical ranges of their estimates and learn how to reduce these ranges
The PSP2 and PSP3 Additions

PSP1 is augmented to include

• personal design and code reviews
• yield and cost of quality measures
• design completion criteria
• design templates

With PSP3, engineers see how to use the PSP for larger scale work.

• cyclic development
• issue tracking log
PSP2 and PSP3 Lessons

With PSP2, engineers use their historical data to improve the quality of the program modules they produce.
- they measure the efficiency of their defect removal methods
- they use various process quality measures, including yield, COQ (cost of quality), and A/FR (appraisal/failure ratio)

With PSP3, engineers learn how to adjust their personal processes for different types of work.
The Basic PSP Elements

A process script

A project plan summary form

A time recording log

A defect reporting log

A defect type standard
Process Script

Scripts guide the engineers through the process.
• purpose
• inputs required
• process phases
• exit criteria

<table>
<thead>
<tr>
<th>Phase Number</th>
<th>Purpose: To guide you in developing module-level programs.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Inputs Required</td>
</tr>
<tr>
<td></td>
<td>• Problem description</td>
</tr>
<tr>
<td></td>
<td>• PSP Project Plan Summary form</td>
</tr>
<tr>
<td></td>
<td>• Historical estimated and actual size and time data</td>
</tr>
<tr>
<td></td>
<td>• Time and Defect Recording Logs</td>
</tr>
<tr>
<td></td>
<td>• Defect Type Standard</td>
</tr>
<tr>
<td></td>
<td>• Stop watch (optional)</td>
</tr>
<tr>
<td>1</td>
<td>Planning</td>
</tr>
<tr>
<td></td>
<td>• Produce or obtain a requirements statement.</td>
</tr>
<tr>
<td></td>
<td>• Estimate the total new and changed LOC required and the prediction interval.</td>
</tr>
<tr>
<td></td>
<td>• Estimate the required development time and the prediction interval.</td>
</tr>
<tr>
<td></td>
<td>• Produce a schedule plan (if more than a couple days)</td>
</tr>
<tr>
<td></td>
<td>• Enter the plan data in the Project Plan Summary form.</td>
</tr>
<tr>
<td></td>
<td>• Complete the Time Recording Log.</td>
</tr>
<tr>
<td>2</td>
<td>Development</td>
</tr>
<tr>
<td></td>
<td>• Design the program, using design templates where appropriate.</td>
</tr>
<tr>
<td></td>
<td>• Review the design and fix and log all defects found.</td>
</tr>
<tr>
<td></td>
<td>• Implement the design.</td>
</tr>
<tr>
<td></td>
<td>• Review the code and fix and log all defects found.</td>
</tr>
<tr>
<td></td>
<td>• Compile the program and fix and log all defects found.</td>
</tr>
<tr>
<td></td>
<td>• Test the program and fix and log all defects found.</td>
</tr>
<tr>
<td></td>
<td>• Complete the Time Recording Log.</td>
</tr>
<tr>
<td>3</td>
<td>Postmortem</td>
</tr>
<tr>
<td></td>
<td>Complete the Project Plan Summary form with actual time, defect, and size data.</td>
</tr>
<tr>
<td></td>
<td>Exit Criteria</td>
</tr>
<tr>
<td></td>
<td>• A thoroughly tested program</td>
</tr>
<tr>
<td></td>
<td>• Completed Project Plan Summary with estimated and actual data</td>
</tr>
<tr>
<td></td>
<td>• Completed design templates</td>
</tr>
<tr>
<td></td>
<td>• Completed Design Review Checklist and Code Review Checklist</td>
</tr>
<tr>
<td></td>
<td>• Completed Test Report Template</td>
</tr>
<tr>
<td></td>
<td>• Complete PIP forms</td>
</tr>
<tr>
<td></td>
<td>• Completed Defect and Time Recording Logs</td>
</tr>
</tbody>
</table>
Project Plan Summary

The project plan summary form holds:

- project plan data
- actual project results
  - size
  - times
  - defect data
- cumulative data on all PSP projects to date
**Time Recording Log**

Time spent working on each PSP phase is recorded.

- start time
- stop time
- interrupt time
- phase
- comments

<table>
<thead>
<tr>
<th>Date</th>
<th>Start</th>
<th>Stop</th>
<th>Interruption Time</th>
<th>Delta Time</th>
<th>Phase</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/1</td>
<td>8:00</td>
<td>8:16</td>
<td>5</td>
<td>11</td>
<td>Plan</td>
<td>Estimated time</td>
</tr>
</tbody>
</table>
Defect Recording Log

Information on each defect found in reviews, compiling, and test is recorded.

- Number
- Type
- Phase injected
- Phase removed
- Find/fix time
- Description

<table>
<thead>
<tr>
<th>Date</th>
<th>Number</th>
<th>Type</th>
<th>Inject</th>
<th>Remove</th>
<th>Fix Time</th>
<th>Fix Defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/1</td>
<td>1</td>
<td>40</td>
<td>code</td>
<td>compile</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>code</td>
<td>compile</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Defect Types
10 Documentation
20 Syntax
30 Build, Package
40 Assignment
50 Interface
60 Checking
70 Data
80 Function
90 System
100 Environment
The PSP Process Flow

Requirements

PSP Process
- Planning
- Development
  - Design
  - Design review
  - Code
  - Code review
  - Compile
  - Test
- Postmortem

Process scripts

Time and defect logs

Finished product

Project plan summary

Project and process data summary report
PSP Planning

In the PSP, engineers use their personal data to make plans.

Planning consists of
• size estimating
• resource estimating
• schedule estimating

In the PSP, lines of code (LOC) are used as the size measure
• Other measures could be used.
The Project Planning Framework

Customer need → Define requirements

- Produce conceptual design
- Estimate size
- Estimate resources

Size database

- Product schedule
- Develop product

- Process analysis
- Resources available

- Size, resource schedule data
- Productivity database

Customer → Product delivery

- Items
- Tasks

Product delivery → Tracking reports

© 1998 Carnegie Mellon University
Why Estimate Size?

To make better plans
• to better size the job
• to divide the job into separable elements

To assist in tracking progress
• can judge when job scope changes
• can better measure the work

Value for the PSP
• learn estimating methods
• build estimating skills
Size Estimating Principles - 1

Estimating is an uncertain process.
• no one knows how big the product will be
• the earlier the estimate, the less is known
• estimates can be biased by business and other pressures

Estimating is an intuitive learning process.
• ability improves with experience
• some people will be better at estimating than others
Size Estimating Principles - 2

The estimating objectives are to
• make consistent estimates
• understand estimate variability
• balance under and over estimates

The principal advantages of using a defined estimating method are
• You have a known practice you can improve.
• It provides a framework for gathering data.
• By using a consistent method and historical data, your estimates will get more consistent
Size Estimating Proxies

A proxy is a substitute

A suitable size estimating proxy will help visualize ill-defined products early in development.

Potential proxies are
• functions, procedures, and methods
• function points
• objects
• report pages, files, screens
Objects as Proxies - 1

Objects make good proxies because

- numbers of objects correlate reasonably well with development hours
- object lines of code (LOC) correlate very closely with development hours
- historical object LOC data can be obtained and used
- using these historical data, object LOC can readily be estimated
Objects as Proxies - 2

When objects are selected as application entities, they can be visualized early in development.

Functions and procedures can often be estimated in the same way.

Objects, functions, procedures, and their LOC can be automatically counted.
Object LOC Correlation with Development Hours
The PROBE Estimating Method

Conceptual Design

Identify and Size Objects
- Number of Methods
- Object Type
- Relative Size
- Reuse Categories

Estimate Other LOC

Estimate Program Size

Calculate Prediction Interval

Size Estimate

Estimate Resources

Calculate Prediction Interval

Resource Estimate
To Make Size Estimates, You Need Several Items

Data on historical objects, divided into types

Estimating factors for the relative sizes of each object type

Regression parameters for computing new and changed LOC from
• estimated object LOC
• LOC added to the base
• modified LOC
## C++ Object Size Ranges

<table>
<thead>
<tr>
<th>Type</th>
<th>VS</th>
<th>S</th>
<th>M</th>
<th>L</th>
<th>VL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculation</td>
<td>2.34</td>
<td>5.13</td>
<td>11.25</td>
<td>24.66</td>
<td>54.04</td>
</tr>
<tr>
<td>Data</td>
<td>2.60</td>
<td>4.79</td>
<td>8.84</td>
<td>16.31</td>
<td>30.09</td>
</tr>
<tr>
<td>I/O</td>
<td>9.01</td>
<td>12.06</td>
<td>16.15</td>
<td>21.62</td>
<td>28.93</td>
</tr>
<tr>
<td>Logic</td>
<td>7.55</td>
<td>10.98</td>
<td>15.98</td>
<td>23.25</td>
<td>33.83</td>
</tr>
<tr>
<td>Set-up</td>
<td>3.88</td>
<td>5.04</td>
<td>6.56</td>
<td>8.53</td>
<td>11.09</td>
</tr>
<tr>
<td>Text</td>
<td>3.75</td>
<td>8.00</td>
<td>17.07</td>
<td>36.41</td>
<td>77.66</td>
</tr>
</tbody>
</table>
Estimate Program Size

Total program size consists of
• newly developed code (adjusted with the regression parameters)
• reused code from the library
• base code from prior versions, less deletions

Newly developed code consists of
• additions to the base
• newly developed objects
• base LOC that are modified
The Prediction Interval

The prediction interval provides a likely range around the estimate.

- a 90% prediction interval gives the range within which 90% of the estimates will likely fall
- it is not a forecast, only an expectation
- it only applies if the estimate behaves like the historical data

The prediction interval is calculated from the same data used to calculate the regression parameters.
The Resource Planning Process

Start with a size estimate

Identify available data

Use regression when you have 3+ sets of data that correlate

Use data for estimated LOC to actual hours where available

Calculate the prediction interval
Schedule Estimating

To make a schedule you need 3 things
• the estimated direct project hours
• a calendar of available direct hours
• the order in which the tasks will be done

You then need to
• estimate the hours needed for each task
• spread these hours over the calendar of available hours
The PSP Quality Strategy - 1

In the PSP, defects are the basic quality measure.

Note that defects are not important to the user as long as they do not

• affect operations
• cause inconvenience
• cost time or money
• cause loss of confidence in the program’s results
The PSP Quality Strategy - 2

Low defect content is an essential prerequisite to a quality software process.
• Experienced software engineers typically inject around 100 defects per KLOC.
• Low defect products can best be assured at the PSP level.

This is where the defects are injected and this is where the engineers should
• remove them
• determine their causes
• learn to prevent them
The PSP Quality Strategy - 3

If you want a quality product out of test, you must put a quality product into test.

- testing removes only a fraction of the defects
- the more defects in the code entering test, the more defects there are on test exit

To manage defects, they must be addressed where they are injected - by each software engineer.

This requires a comprehensive focus on software quality.
The PSP Quality Strategy - 4

Data show that it is much more efficient to find defects in reviews than in testing.
- in unit test, typically only about 2 to 4 defects are found per hour
- code reviews typically find about 10 defects per hour
- experienced reviewers can find 70% or more of the defects in a product
- unit test rarely exceeds a 50% yield

PSP data show that reviews find 2 to 5 times as many defects per hour as unit test.
PSP Reviews

In a personal design or code review
• Professionals privately review their products.
• Their objective is to find all defects before the first compile and test.
• Reviews are most effective when structured and measured.

Reviews can be used for requirements, designs, and code.
Review Yield - 1

Yield

- a measure of process quality
- the percent of defects in the product at review time that were found by the review
- a measure of the effectiveness of a process step
  - design and code reviews
  - the overall process - prior to test
  - the development process - including test

Yield(for a phase or the entire process) = 100*(defects found)/(defects found + not found)
Review Yield - 2

Yield cannot be calculated until all defects have been found through test and product use.

Yield can be useful early in the process if all or most defects are counted.
  • design and code review defects
  • compile defects
  • unit test defects

Using process data, control parameters can help to ensure high yield reviews.
The Cost of Quality (COQ) - 1

Failure costs
• repair, rework, and scrap
• PSP failure costs are compile and test time

Appraisal costs
• costs of inspecting for defects
• PSP appraisal costs are design review and code review time

Prevention costs are finding and resolving defect causes.
The Cost of Quality (COQ) - 2

A useful PSP measure is the ratio of appraisal to failure costs (A/FR).

- \( A/FR = \frac{\text{appraisal COQ}}{\text{failure COQ}} \)
- A/FR measures process quality.

A/FR experience

- If measured, the A/FR of most software organizations would be near zero
- In the PSP, A/FR should exceed 2.0
- High A/FR is associated with low numbers of test defects and high product quality
Messages to Remember

1. A defined and measured process provides a repeatable basis for improvement.

2. The PROBE method provides a statistically sound framework for planning.

3. The PSP quality strategy will help engineers produce high quality products.