Personal Software Process (PSP)

- Application of CMM principles to individuals
- Developed by Watts Humphrey of the Software Engineering Institute (SEI) in the early 1990s
  - Extensive supporting materials: books, courses, forms, exercises
- Validated by data from numerous projects
  - 58% reduction in defects/KLOC (development)
  - 72% reduction in defects/KLOC (testing)
  - 21% improvement in productivity
- Complemented by Team Software Process (TSP)
- Strict waterfall plus process monitoring and improvement
PSP and CMM

• Complementary
  – CMM is top-down - management oriented
  – PSP is bottom-up - engineer oriented

• Level 2
  – Software configuration management
  – Software quality assurance
  – Software subcontract management
  – Software project tracking and oversight
  – Software project planning
  – Requirements management

• Level 3
  – Peer reviews
  – Intergroup coordination
  – Software product engineering
  – Integrated software management

• Level 4
  – Training program
  – Organization process definition
  – Organization process focus

• Level 5
  – Software quality management
  – Quantitative process management

• Level 5
  – Process change management
  – Technology change management
  – Defect prevention
Overview

• Disciplined personal framework for developing software
  – 50-5000 LOC projects
• Metrics, forms, and scripts
• Produce low-defect products on schedule and within planned costs
• Manage quality, analyze results, improve process
Assumptions/Principles

- Every engineer is different. To be most effective, engineers must plan their work, and they must base their plans on their own personal data.
- To consistently improve their performance, engineers must use well-defined and measured processes.
- To produce quality products, engineers must feel personally responsible for the quality.
- It costs less to find and fix defects earlier in a process than later.
- The right way is always the fastest and cheapest way to do a job.
Overall Approach

• Experienced programmers inject one defect per 7-10 lines of code
• People tend to make the same mistakes repeatedly
• To improve your organization's performance
  – Record data on defects; review data; make process changes to eliminate causes
  – Spend more up front time (design and detection activities)
Process Structure

- Scripts
  - Planning
    - Design
    - Design Review
    - Code
    - Code Review
    - Compile
    - Test
    - Postmortem
  - Finished Product
  - Requirements
    - Plans
      - Results
        - Logs
          - Plan summary
            - Project and process data summary report

- Time Defects
# PSP Phases

<table>
<thead>
<tr>
<th>Phase</th>
<th>Emphasis</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Personal Management</td>
<td>Current process plus basic measures: development time, defects injected and removed; process: planning, development, analysis</td>
</tr>
<tr>
<td>0.1</td>
<td></td>
<td>Coding standards, process improvement proposal form, size measurements</td>
</tr>
<tr>
<td>1</td>
<td>Personal Planning</td>
<td>PROBE; Size estimation, time estimates, test report</td>
</tr>
<tr>
<td>1.1</td>
<td></td>
<td>Task planning, schedule planning</td>
</tr>
<tr>
<td>2</td>
<td>Personal Quality</td>
<td>Defect management: code reviews, design reviews</td>
</tr>
<tr>
<td>2.1</td>
<td></td>
<td>Design specification and analysis; defect prevention; process analysis; process benchmarks</td>
</tr>
<tr>
<td>3</td>
<td>Scaling Up</td>
<td>Cyclic development</td>
</tr>
</tbody>
</table>
PSP0

- Personal measurement
- Forms and scripts
- Time, defects injected and removed
- Phases: planning, development, postmortem
- PSP0.1: add in coding standards, size measurement, and process improvement proposal
PSP1

- Personal planning
- PROBE estimation; confidence intervals
- PSP1.1: schedule and task planning
<table>
<thead>
<tr>
<th>Phase Number</th>
<th>Purpose</th>
<th>To guide you in developing module-level programs</th>
</tr>
</thead>
</table>
| Entry Criteria | • Problem description  
• PSP1 Project Plan Summary form  
• Size Estimating Template  
• Historical estimate and actual size data  
• Time and Defect Recording Logs  
• Defect Type Standard  
• Stop watch (optional) | |
| 1 | Planning | • Produce or obtain a requirements statement.  
• *Use the PROBE method to* estimate the total new and changed LOC required.  
• *Complete the Size Estimate Template.*  
• Estimate the required development time.  
• Enter the plan data in the Project Plan Summary form.  
• Complete the Time Recording Log. | |
| 2 | Development | • Design the program.  
• Implement the design.  
• Compile the program and fix and log all defects found.  
• Test the program and fix and log all defects found.  
• Complete the Time Recording Log. | |
| 3 | Postmortem | Complete the Project Plan Summary form with actual time, defect, and size data. | |
| Exit Criteria | • A thoroughly tested program  
• Completed Project Plan Summary form with estimated and actual data  
• *Completed Size Estimating Template*  
• *Completed Test Report Template*  
• Completed PIP forms  
• Completed Defect and Time Recording Logs | |
PSP2

• Personal quality
• Defect management: data, review checklists
• PSP2.1: design specification, defect prevention, process analysis, process benchmarks
PSP3

- Scaling up
- Cyclic development
- Design verification; process definition principles
- Subsumed by TSP
Overall PSP Strategy

1. Gather data
2. Estimate and plan
3. Manage defects
4. Manage yield
5. Control cost of quality
1. Gathering Data

- Measurements taken
  - Time in each process activity (and for interrupts)
  - Defects introduced and removed for each activity
  - Developed product size (LOC)
    - Base, added, modified, deleted, new and changed, reused, new reuse, total

- Metrics computed
  - Size and time estimating error
  - Cost-performance index
  - Defect
    - Injected and removed per hour
    - Density
  - Process yield
  - Appraisal and failure cost of quality
  - Appraisal to failure ratio
2. Estimate and Plan

- PROBE - proxy based estimation method
- PSP proxies: functions and object
  - Others include function points, screens, reports, sections of text
- Linear regression on at least 3 prior projects
- Goal is to improve estimates over time
  - PSP students improved their size estimates from 31% (within 20%) to 42% between programs one and ten
  - Improved time estimates from 33% (within 20%) to 49%
Example PROBE Data (C++)

<table>
<thead>
<tr>
<th>Category</th>
<th>Very Small</th>
<th>Small</th>
<th>Medium</th>
<th>Large</th>
<th>Very Large</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>
Size Categories (SEI)

- **Base**: When an existing product is enhanced, base LOC is the size of the original product version before any modifications are made.
- **Added**: Code written for a new program or added to an existing base program.
- **Modified**: LOC in an existing (Base) program that are changed.
- **Deleted**: LOC in an existing (Base) program that are deleted.
- **New and Changed**: When engineers develop software, it takes them much more time to add or modify a LOC than it does to delete or reuse one. Thus, in the PSP, engineers use only the **Added** or **Modified** code to make size and resource estimates. This code is called the **New and Changed** LOC.
- **Reused**: Code taken from a reuse library and used, without modification, in a new program. Reuse does not count the unmodified base code retained from a prior program version and it does not count any code that is reused with modifications.
- **New Reuse**: LOC that an engineer develops and contributes to the reuse library.
- **Total**: Total size of a program, regardless of its source (= **Base** - **Deleted** + **Added** + **Reuse**).
3. Manage Defects

- Record, for each defect
  - Activity (phase) during which defect was injected and removed
    - Planning, design, design review, code, code review, compile, test
  - Defect type (next slide)
  - Fix time
  - Description

- Students reduced defect rates from 116/KLOC to 49/KLOC between programs one and ten
  - Standard deviation also reduced
# Defect Types

<table>
<thead>
<tr>
<th>Type Number</th>
<th>Type Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Documentation</td>
<td>comments, messages</td>
</tr>
<tr>
<td>20</td>
<td>Syntax</td>
<td>spelling, punctuation, types, instruction formats</td>
</tr>
<tr>
<td>30</td>
<td>Build, package</td>
<td>change management, library, version control</td>
</tr>
<tr>
<td>40</td>
<td>Assignment</td>
<td>declaration, duplicate name, scope, limits</td>
</tr>
<tr>
<td>50</td>
<td>Interface</td>
<td>procedure calls and references, I/O, user format</td>
</tr>
<tr>
<td>60</td>
<td>Checking</td>
<td>error messages, inadequate checks</td>
</tr>
<tr>
<td>70</td>
<td>Data</td>
<td>structure, content</td>
</tr>
<tr>
<td>80</td>
<td>Function</td>
<td>logic, pointers, loops, recursion, computations, function defects</td>
</tr>
<tr>
<td>90</td>
<td>System</td>
<td>configuration, timing, memory</td>
</tr>
<tr>
<td>100</td>
<td>Environment</td>
<td>design, compile, test, or other support-system problems</td>
</tr>
</tbody>
</table>
Defects per KLOC Trend
(Humphrey - Fig. 4)

Observations
- Standard deviation also reduced
- Student programmers
- Hawthorn effect?
- Compilation defects fall faster
Question

• Would you rather have your testing group uncover a lot of failures or a few?
Would you rather have your testing group uncover a lot of failures or a few?
4. Manage Yield

• Yield is PSP's principle quality measure
• If it is costly to find a defect during testing, then you need to find it earlier (during review)
  – (Or not insert it in the first place)
• Hold review before compilation
  – (But aren't compilers cheaper than programmers?)
  – (And desk check every new compilation)
Yield

- *Yield*: % defects found and fixed before compilation
  - Engineers review code before first compile
  - 9% of "syntax" error get by compiler
  - Defects found at compile time correlate with defects found during test \(r = .71\)
  - Strong correlation between defects found during test and customer failures \(r = .91\)
- Introduction of design and code reviews strongly improves yield
Yield versus Program Number
(Humphrey - Fig. 7)

Observations
- Program 7 introduced reviews
5. Control Cost of Quality

• Appraisal cost
  – Time spent in design and code reviews

• Failure cost
  – Time spent in compile and test

• Prevention costs
  – Prototyping, formal specification
  – Not part of PSP

• Appraisal to failure ratio (A/FR)
  – Raise until quality is sufficient then gradually lower
  – Initial target at least two
Total Defects per KLOC versus A/FR

(Humphrey - Fig. 9)

Observations
• Little improvement after 3:1
• Enables control of the productivity / quality tradeoff
How Much Time should you Spend in Reviews?
How Much Time should you Spend in Reviews?

• Spend as much time reviewing as is required to detect and remove all defects injected during the activity being reviewed
• Depends on the rates of fault injection and removal per time unit
• This means that you had better measure these rates
• PSP measurements on students indicate that they should spend 59% as much time reviewing as injecting for design activities and 65% for code
Another Answer

• PSP rule of thumb is to find twice as many problems during code review as you do during testing

• So if for module A, you found 15 during review and 45 during testing, you need to increase your review time by a factor of six!

  \[ 15 \times 6 = 90 = 2 \times 45 \]
Design

• PSP does not prescribe a design method
  – Instead, it emphasized design completion
  – So it recommends making sure of the following

• Example schema
  – External static
    • Function interfaces: signatures, inheritance
  – External dynamic
    • Operational scenarios, call/return
  – Internal static
    • Attributes, constraints
  – Internal dynamic
    • State machines, response time, interrupts
PSP Results

• Estimation improvement
  – Reduced variance leads to better scheduling and staffing

• Reduced compile and test defects
  – Correlated with reduced customer-detected failures

• Mild productivity improvement
PSP Benefits

• Increases personal commitment by investing each engineer with process responsibility
• Assists engineers in making accurate plans
• Provides steps engineers can take to improve personal and project quality
• Sets benchmarks to measure personal process improvements
• Demonstrates the impact of process changes on an engineer's performance