Lecture 8: *The Personal Software Process*
PSP: the background

**CMMI**: Capability Maturity Model Integration  
(originally: CMM)

From late 1980s, Software Engineering Institute  
(At Carnegie-Mellon University,  
funded by the US Department of Defense)

Goal: assess the maturity of the software process of an organization, especially its reproducibility

Five levels of maturity:

- Initial
- Managed
- Defined
- Quantitatively managed
- Optimizing
## CMMI: management visibility

<table>
<thead>
<tr>
<th>Level</th>
<th>Process Characteristics</th>
<th>Management Visibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimizing</td>
<td>Focus is on continuous quantitative improvement</td>
<td><img src="image1" alt="Diagram" /></td>
</tr>
<tr>
<td>Quantitatively Managed</td>
<td>Process is measured and controlled</td>
<td><img src="image2" alt="Diagram" /></td>
</tr>
<tr>
<td>Defined</td>
<td>Process is characterized for the organization and is proactive</td>
<td><img src="image3" alt="Diagram" /></td>
</tr>
<tr>
<td>Managed</td>
<td>Process is characterized for projects and is often reactive</td>
<td><img src="image4" alt="Diagram" /></td>
</tr>
<tr>
<td>Initial</td>
<td>Process is unpredictable, poorly controlled, and reactive</td>
<td><img src="image5" alt="Diagram" /></td>
</tr>
</tbody>
</table>
Team Software Process
Personal Software Process

Outgrowth of CMMI work

Directs work of teams and individuals for higher quality

PSP is part of TSP
About this presentation

This lecture describes the PSP as seen by its authors.

It does not necessarily imply endorsement of every idea.

The symbol indicates points that seem arguable.
Management support

The initial TSP objective is to convince management to let the team be self-directed, meaning that it:

- Sets its own goals
- Establishes its own roles
- Decides on its development strategy
- Defines its processes
- Develops its plans
- Measures, manages, and controls its work
Management support

Management will support you as long as you:

- Strive to meet their needs
- Provide regular reports on your work
- Convince them that your plans are sound
- Do quality work
- Respond to changing needs
- Come to them for help when you have problems
Management support

Management will agree to your managing your own work as long as they believe that you are doing a superior job.

To convince them of this, you must:

- Maintain and publish precise, accurate plans
- Measure and track your work
- Regularly show that you are doing superior work

The **PSP** helps you do this
PSP principles

The quality of a software system is determined by the quality of its worst component.

The quality of a component is governed by the individual who developed it, and especially by the quality of the process he or she used to develop it.

Every software professional is responsible for his or her personal process. Key to quality is each individual developer’s skill, commitment, and personal process discipline.
PSP essential practices

- Measure, track, and analyze your work
- Learn from your performance variations
- Incorporate lessons learned into your personal practices
What does a PSP provide?

A stable, mature PSP allows you to

- Estimate and plan your work
- Meet your commitments
- Resist unreasonable commitment pressures

You will also

- Understand your current performance
- Improve your expertise as a professional
As a personal process, PSP includes:

- Defined steps
- Forms
- Standards
- A measurement and analysis framework for characterizing and managing your personal work
- A defined procedure to help improve your personal performance
The PSP process flow

Requirements

Planning
Design
Code
Compile
Test
Postmortem

Finished product

Logs

Scripts

guide

Project summary

Project and process data summary report
A progressive approach

PSP is introduced in six upward-compatible steps

At each step:

- Write one or more modules
- Gather and analyze data on your work
- Use results to improve your personal performance
The steps

- **TSP**
  - Team development

- **PSP2**
  - Code reviews
  - Design reviews

- **PSP1**
  - Size estimating
  - Test report

- **PSP0**
  - Current process
  - Time recording
  - Defect recording
  - Defect type standard

- **PSP0.1**
  - Coding standard
  - Size measurement
  - Process improvement proposal (PIP)

- **PSP2.1**
  - Design templates

- **PSP1.1**
  - Task planning
  - Schedule planning
Goals at each level

PSP0: Establish a measured performance baseline

PSP1: Make size, resource, and schedule plans

PSP2: Practice defect and yield management
Objective:

- Demonstrate use of defined process for small programs
- Incorporate basic measurements in process
- Minimize changes to your personal practices
PSP0 setup

PSP0 is a simple, defined, personal process:

- Make a plan
- Use your current design and development methods to produce a small program
- Gather time and defect data on your work
- Prepare a summary report
The six phases of PSP

1. Plan
   Produce plan for developing program from requirements

2. Design
   Produce design specification for the program.

3. Code
   Turn design into executable code (In Eiffel, 2 & 3 are one step)

4. Compile
   Translate into executable code

5. Test
   Verify that code satisfies requirements

6. Postmortem
   Summarize & analyze project data
Phase order

PSP looks like waterfall but is not

Phase order is determined by dependencies:

- Can’t test code before it’s compiled
- Can’t compile before it’s written
- Can’t use design if produced after code is written
- No reason to make a plan after you’re done

Conclusion: start here with a plan
Cyclic process flow

Programs that are large programs or not well understood may require an iterative approach.

In this example, each module is separately coded, compiled, and tested.

The example uses PSP0 phases and 2 code-compile-test cycles.
Cyclic process flow

There can be more than 2 cycles

Part size is key factor for determining cycles:
- Line of code: too small
- Program: usually too large

Typical: one or more classes or features

Determine what works for you
Objective: help you to
- Measure size of programs that you produce
- Perform size accounting for these programs
- Make accurate and precise size measurements
Process measurement

To be useful, measurements should be
- Gathered for a specific purpose
- Explicitly defined
- Properly managed
- Properly used

We measure to
- Understand and manage change
- Predict or plan
- Compare one product, process, or organization with another
- Determine adherence to standards
- Provide a basis for control
Measurement objectives

Measurements only produce numbers

To be useful, they must

- Relate to business objectives
- Be properly interpreted
- Lead to appropriate action

If the business purposes for the measurements are not understood

- The wrong data may be gathered
- Data may not be properly used
PSP measurements

Basic PSP data:
- Program size
- Time spent by phase
- Defects found and injected by phase

On every item, gather both actual and estimated data

Measures derived from these data:
- Support planning
- Characterize process quality
Objective:
Establish orderly & repeatable procedure for size estimation

New process elements:
- PROBE size estimating method & template
- Test report template
Estimating with PROBE

Stands for PROxy Based Estimating

Uses proxies to estimate program size and development time

A good proxy helps make accurate estimates
The PROBE estimating method

Start

Conceptual design

Identify and size the proxies
Number of items  Part Type  Relative size  Reuse categories

Estimate other element sizes

Estimate program size

Calculate prediction interval

Size estimate and range

Estimate resources

Calculate prediction interval

Resource estimate and range
Conceptual design

Conceptual design relates the requirements to the parts needed to produce the program

Parts categories:
- **Reused**: Can be used as-is
- **Base**: Exists, requires modifications
- **Added**: needs to be developed
Sizing parts

Reused part: Use actual size

Added part: define proxy

- Identify part type, e.g. parsing, GUI, network…
- Estimate number of items, e.g. routines
- Estimate relative size, i.e. very small, small, medium, large, or very large
- Find size of an item of this part type and relative size in the relative size table
- Estimated size = item size * number of items

Base part: start from actual size; estimate additions, deletions, modifications
Objective: introduce & practice methods for
- Making resource & schedule plans
- Tracking your performance against them
- Judging likely project completion dates

Two new process elements:
- Task planning template
- Schedule planning template

Typically used for projects that take several days or weeks
Objective: introduce

- Design & code reviews
- Methods for evaluating and improving quality of reviews

Two key capabilities added at this level:
- Design and code reviews
- Quality planning

Two new process elements, separate:
- Design review checklist
- Code review checklist
PSP2 introduces quality planning. This involves estimating:

- Total number of defects that will be injected
- Number of defects injected & removed in each process phase
- Amount of time for design and code reviews

& adjusting these parameters to ensure high-quality result
Arguments for reviews over tests

In testing, you must

- Detect unusual behavior
- Figure out what the test program was doing
- Find where the problem is in the program
- Figure out which defect could cause such behavior

This can take a lot of time

With reviews you

- Follow your own logic
- Know where you are when you find a defect
- Know what the program should do, but did not
- Know why this is a defect
- Are in a better position to devise a correct fix
PSP review process principles

Defined review process: guidelines, checklists, standards.

Goal is to find every defect before first compile/test

To meet it, you must:

- Review before compiling or testing
- Use coding standards
- Use design completeness criteria
- Measure and improve your review process
- Use a customized personal checklist
Code review checklist

Reviews are most effective with personal checklist customized to your own defect experience:

- Use your own data to select the checklist items
- Gather and analyze data on the reviews
- Adjust the checklist with experience

Do the reviews on a printed listing, not on screen

The checklist defines steps and suggests their order:

- Review for one checklist item at a time
- Check off each item as you complete it
Design review principles

In addition to reviewing code, you should also review your designs.

Requires that you
- Produce designs that can be reviewed
- Follow an explicit review strategy
- Review design in stages
- Verify that logic correctly implements requirements
Objective: introduce

- Additional measures for managing process quality
- Design templates that provide an orderly framework and format for recording designs

New process elements:

- Design review script
- Design review checklist
- Operational specification template
- Functional specification template
- State specification template
- Logic specification template
PSP: an assessment

Ignore technology assumptions (strict design-code-compile-test cycle) which is not in line with today’s best practices. Retain emphasis on professional engineer’s approach:

- **Plan**
- **Record** what you do both qualitatively and quantitatively:
  - Program size
  - Time spent on parts and activities
  - Defects
- **Think** about your personal process
- **Improve** your personal process

Tool support, integrated in development environment, is essential.