Introduction to the Personal Software Process

SWENET PRO2 Module

Developed with support from the National Science Foundation

Overview

- Process Fundamentals
- PSP Concepts and Structure
- PSP Planning and Measurement
- PSP Quality Management
- PSP Results
- References

Software Processes 1

- A process is a series of steps involving activities, constraints and resources that produce an intended output of some kind.
- A software process (or a software engineering process) guides and supports the development of a software product.
- In the last decade there has been a great of deal of resources devoted to the definition, implementation, and improvement of software development processes.
- The list of references at the end of this presentation refer to papers that provide background and discussion for further study of software process features and issues.

Software Processes 2

- A "defined" software process would have a documented description of its features which would typically include the following:
  - scripts that define the process steps
  - standards and procedures for carrying our process steps
  - forms and templates for collecting process data and documenting process outcomes

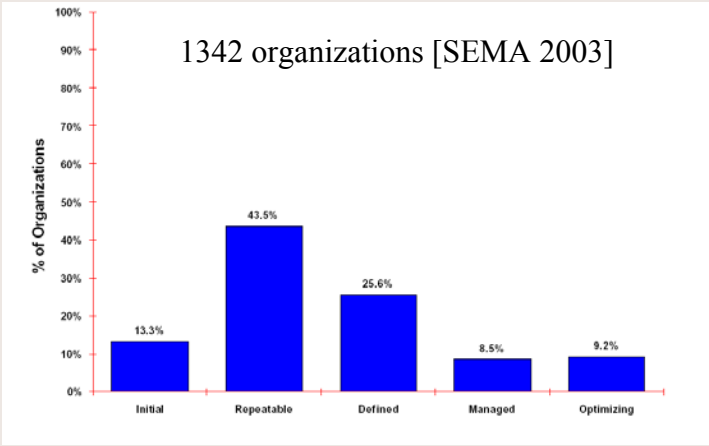
A Process Framework

- In the late 1980s, the Software Engineering Institute (SEI), Carnegie Mellon University, developed the SW-CMM (Software Capability Maturity Model) to help organizations build effective software engineering processes [Paulk 1993]. It has been widely adopted in industry, primarily by large software development organizations.
- The next slide describes the five-level model encompassing good engineering and management practices.
  - The model is based on effective software engineering practices called Key Process Areas (KPAs).
- Many software development organizations have been assessed using the SW-CMM framework. Results of such assessments are shown in a later slide.

SW-CMM Levels

Level	Focus	Decription
5: Optimizing	Continuous Process Improvement	Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.
4: Managed	Product and Process Quality	Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
3: Defined	Engineering Process	The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
2: Repeatable	Project Management	Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
1: Initial	No Focus	Project success primary depends on individuals and their heroics.

SW-CMM Assessment



PSP Concepts

- The Personal Software Process[SM] (PSP[SM]) is software process developed at the SEI to address some of the SW-CMM practices at the level of the individual programmer [Humphrey 1995].
- The PSP provides an incremental approach that helps engineers develop an individual "level 5" process.
- In PSP training, engineers use these processes to
  - do 10 software development exercises
  - analyze the data on their work
  - improve their performance
- PSP embodies a "self-convincing" philosophy.
  - That is, engineers can use their own data to determine the value of the PSP.
- PSP training is organized into a hierarchical, incremental model.

Software Engineering and the PSP

- The PSP offers an opportunity and framework for incorporating software engineering best practices into an individual engineer's process.
  - The PSP address's 12 of the SW-CMM 18 KPA's.
  - For this reason, PSP sometimes referred to as a "Level 5" process.
- Data about PSP training show that students and engineers can improve their planning capabilities and produce higher quality products without an increase in cost.

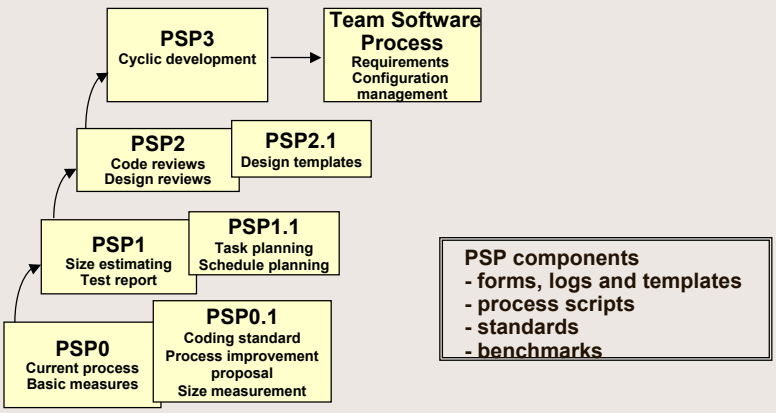
PSP Key Process Areas

Level	Focus	Key Process Area
5: Optimizing	Continuous Process Improvement	• Defect Prevention (PSP) • Technology Change Management (PSP) • Process Change Management (PSP)
4: Managed	Product and Process Quality	• Quantitative Process Management (PSP) • Software Quality Management (PSP)
3: Defined	Engineering Process	• Organizational Process Focus (PSP) • Organizational Process Definition (PSP) • Integrated Software Management (PSP) • Training Program • Software Product Engineering (PSP) • Intergroup Coordination • Peer Reviews (PSP)
2: Repeatable	Project Management	• Requirements Management • Software Project Planning (PSP) • Software Project Tracking and Oversight (PSP) • Software Subcontract Management • Software Quality Assurance • Software Configuration Management

PSP Courses

- The "full" PSP course
  - industry course: 3 intensive weeks (60 hours per week)
  - academic course: 12 to 15 weeks (3 hours per week)
  - includes 10 programming assignments and 5 reports
  - course results similar in industry and academia
  - uses *A Discipline for Software Engineering* [Humphrey 1995]
- A "introductory" PSP course
  - does not teach the complete PSP
  - is often offered as part of a project management course
  - may start in the 1st year of a computing degree
  - uses *Introduction to the Personal Software Process* [Humphrey 1997]

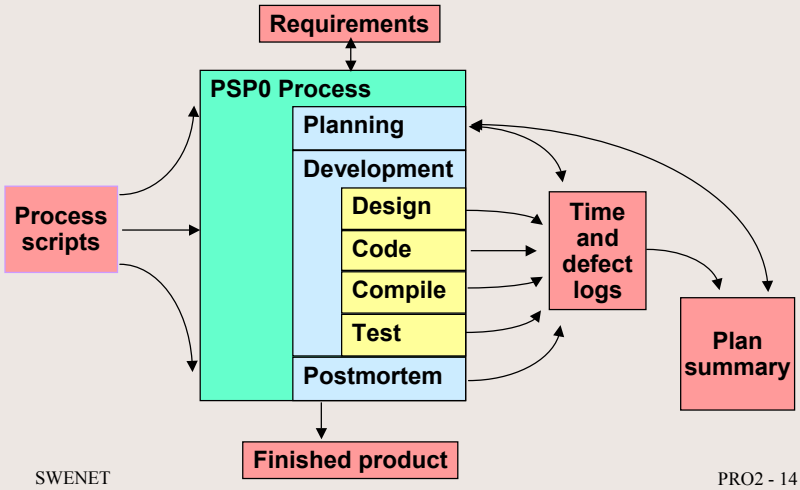
PSP Structure 1



PSP Structure 2

- PSP0: establish a measured performance baseline
- PSP1: make size, resource, and schedule plans
- PSP2: learn defect and yield management
- PSP3: scale up PSP methods to larger projects
- The PSP can be extended to team development of large-scale software systems.
  - –The PSP is a pre-requisite for the Team Software Process(TSP).

The PSP0 Process Flow



PSP0 Process Script

	Purpose:	To guide you in developing module-level programs.
	Inputs Required	Problem description PSP0 project plan summary form Time and defect recording logs Defect type standard Stop watch (optional)
1	Planning	- Produce or obtain a requirements statement. - Estimate the required development time. - Enter the plan data in the project plan summary form. - Complete the time 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 with estimated and actual data - Completed defect and time logs

PSP Planning

- The PSP shows engineers how to estimate and plan their work.
- The keys to making better estimates and plans are to use
  - relevant historical data
  - statistically sound methods
  - a defined estimating and planning process
- As engineers gain experience, they learn to make better estimates and plans.

Measurement in a Defined Process

- Measured historical data is needed for effective planning.
- Measurement tells when and how process tasks are carried out.
- Measured data is used to evaluate and improve a process.
- The PSP uses three types of measures.
  - effort
  - size
  - defects

Effort Measurement

- The PSP measures effort as time in minutes.
  - appropriate for small programs
  - easy to measure precisely
- You keep accurate records of time spent on each programming task.
  - A Time Recording Log is used for this purpose.
- Interruption time is recorded and subtracted from time spent on development tasks.
- Record all time spent on software development.

Time Recording Log 1

Student _____ Date _____
Instructor _____ Class _____
Program _____

Date	Start	Stop	Int	delta t	Activity	Comment

Time Recording Log 2

- Phase
  - note the phase on which you were working
- Date - Enter the current date (e.g., 9/5/03)
- Start - Enter the time in minutes (e.g., 11:25 am) when you start a project phase.
- Interruption time - Enter any time you lost due to interruptions in the start to stop period. (e.g., 15 minute coffee break)
- Stop - Enter the time in minutes when you stop work on a project phase, even if you are not done with that phase.
- Comments - describe
  - the interruption
  - the task you were doing
  - anything else that significantly affects your work

Size Measurement 1

- Size data is used in estimating development time and the expected number of defects.
- There are a number of criteria for good size measures.
  - has good correlation with effort
  - has a precise definition
  - can be counted automatically
  - is suitable for planning
  - is sensitive to language, design, and development method
- A lines of code (LOC) measure satisfies most of the criteria for good size measures.

Size Measurement 2

- The PSP uses LOC for measuring size.
- There are various ways to define and interpret the meaning of a line of code.
  - A *physical LOC* count is the number of non-blank, non-comment lines of source code.
  - A *logical LOC* count is related to a program's logical content, but it depends on the programmer's definition of a logical line of code.
- For simplicity, the PSP uses a coding standard that requires each logical LOC to use one physical line.

Defect Measurement

- A PSP defect is
  - something that must be changed to correct an error made in a software artifact (design, code, etc.)
  - classified according to a defect type standard
- For each defect, students record the defect type, a description of the defect, and the fix time.
  - All changes related to a single error are counted as one defect.
  - Fix time is recorded in the phase in which the defect is removed (e.g, compile or test).
  - A Defect Recording Log is used for this purpose.

Defect Recording Log 1

Student						Date	
Instructor						Class	
Program							

Date	Def. Num.	Type	Phase Injected	Phase Removed	Fix Time	Description

type	code	name	description
doc	10	Documentation	comments, messages
syn	**20**	**Syntax**	**spelling, punctuation, typos, instruction formats, etc.**
bld	30	Build, Package	change management, library, version control
asg	**40**	**Assignment**	**declaration, identifier names, scope, limits**
int	**50**	**Interface**	**procedure calls, context clauses, and references, I/O, user prompts, output labeling**
chk	60	Checking	error messages, inadequate checks and exception handling
dat	70	Data	structure, content
fun	**80**	**Function**	**logic, pointers, loops, recursion, computation, function defects**
sys	90	System	system configuration, timing, memory
env	100	Environment	tool support and operating system problems

Defect Recording Log 2

- Header - enter the name, date, instructor, and program number
- Date - Enter the date when you found and fixed the defect.
- Number - Enter a unique number for this defect. Start each project with 1.
- Type - Enter the defect type from the defect type standard.
- Inject - Enter the phase in which you judge the defect was injected.
- Remove - Enter the phase in which you found and fixed the defect.
- Fix time - Enter the time you took to fix the defect. You may time it exactly or use your best judgment.

Plan Summary Form

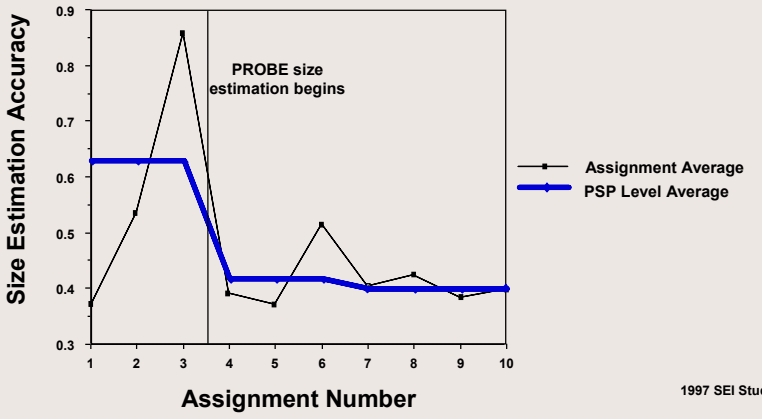
- Measured data is used for planning, tracking and analyzing software development activities.
- The Plan Summary form is used for recording the planning data and summarizing the results.
- Tables 19.1 and 19.2 in [Humphrey 1997] give instructions and an example for completing the form.

Program Size (LOC)	Plan	Actual	To Date	
Total New & Changed				
Maximum Size				
Minimum Size				
Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning				
Design				
Design Review				
Code				
Code Review				
Compile				
Test				
Postmortem				
Total				
Maximum Time				
Minimum Time				
Defects Injected	**Plan**	**Actual**	**To Date**	**To Date %**
Design				
Design Review				
Code				
Code Review				
Compile				
Test				
Total				
Defects Removed	**Plan**	**Actual**	**To Date**	**To Date %**
Design				
Design Review				
Code				
Code Review				
Compile				
Test				
Total				
Summary	**Plan**	**Actual**	**To Date**	
LOC/Hour				
Defects/KLOC				
Pre-Compile Yield				

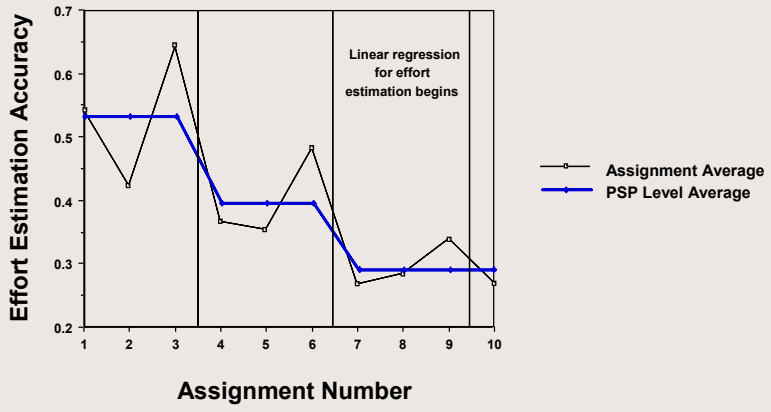
Study of PSP Training

- In 1997, the SEI conducted a study of PSP training [Hayes 1997].
- 23 classes provided data to the SEI
  - instructor training led by SEI personnel
  - academic courses taken by graduate and upper-level undergraduate students
  - industry offerings where SEI and non-SEI instructors taught on-site
- Total of 298 engineers/students

Size Estimating Accuracy



Effort Estimating Accuracy



1997 SEI Study

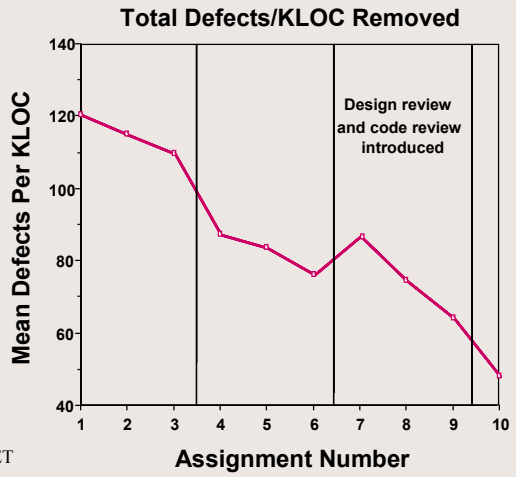
Personal Quality Management

- In the PSP, students conduct personal design and code reviews.
- Engineers use a defined review process.
  - They design their own review checklists.
  - They use a review script and rigorously follow the checklists.
  - They collect data on reviews and used to improve their review process.
- Engineers collect data, compute metrics from the data, analyze the metrics, and use the results to improve the quality of their process and products.

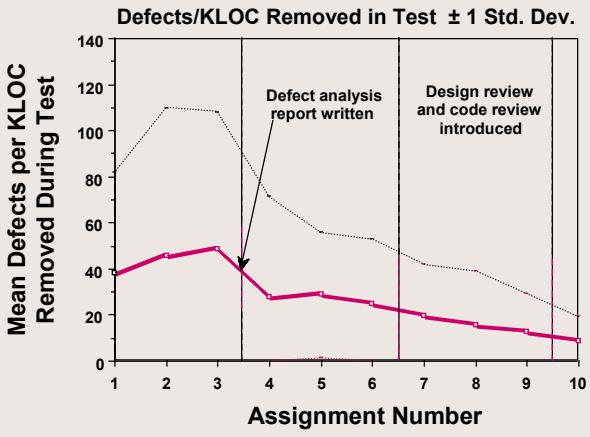
PSP Quality Measures

Quality Measure		Description
Product Quality	Total defects/KLOC	the number of defects found in development, per 1000 lines of code
	Test defects/KLOC	the number of defects found in test, per 1000 lines of code
Process Quality	Yield	the percent of defects found before compile
	Appraisal COQ (Cost of Quality)	the percent of total develoment time spent in design review and code review
	Failure COQ	the percent of total develoment time spent in compile and test
	Total COQ	Appraisal COQ + Failure COQ
	A/F R	$\frac{\text{Appraisal COQ}}{\text{Failure COQ}}$
	Review rate - LOC/hour	the number of lines of code reviewed per hour in a review (code review or design review)
	Defect removal rate - defects/hour	the rate at which defects are removed in a defect removal phase (design review, code review, compile, test)

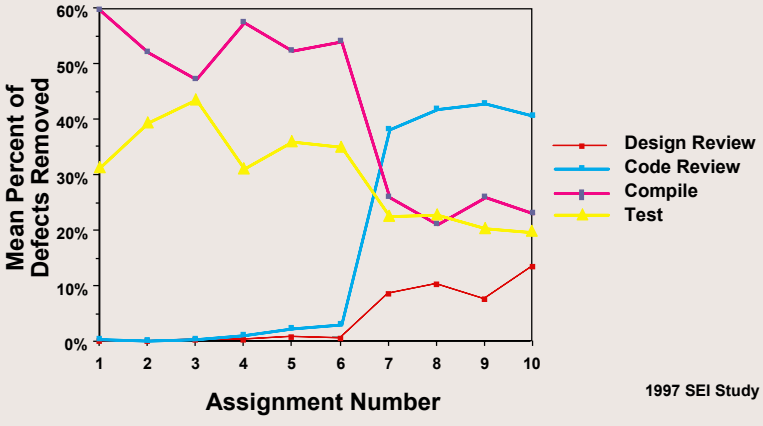
Total Defects/KLOC Removed



Test Defects/KLOC Removed



Defect Removal Comparisons

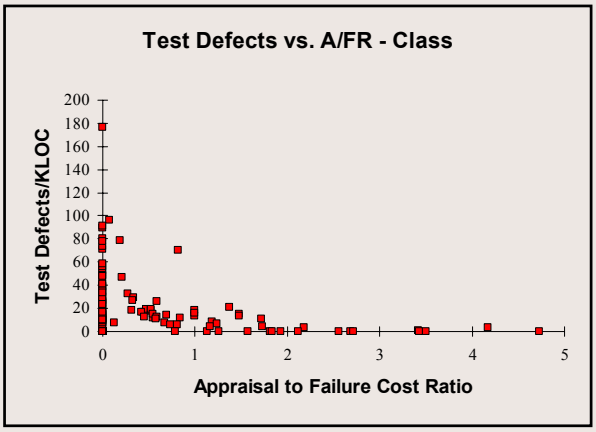


Defect Fix Times

- Defect fix times (in minutes) for an individual PSP student:

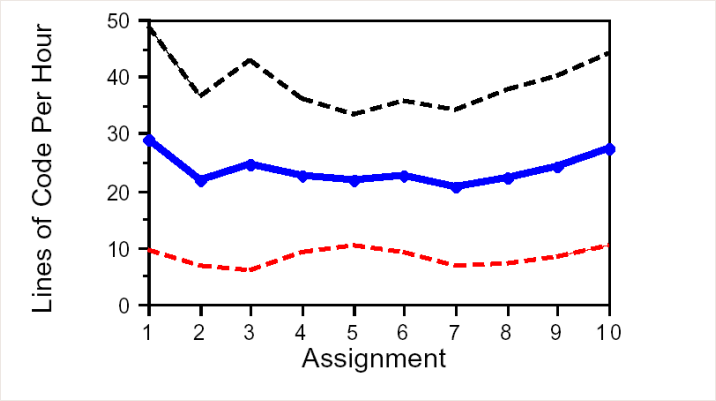
Defects Injected	Defects Found	Design Review	Code Review	Compile	Test	Total
Design	Tot. Fix Time	21	5	3	109	138
	Tot. Defects	17	4	1	6	28
	Avg. Fix Time	1.24	1.25	3.00	18.17	4.93
Code	Tot. Fix Time		32	52	75	159
	Tot. Defects		29	42	15	86
	Avg. Fix Time		1.10	1.24	5.00	1.85
Total	Tot. Fix Time	21	37	55	184	297
	Tot. Defects	17	33	43	21	114
	Avg. Fix Time	1.24	1.12	1.28	8.76	2.61

Test Defects/KLOC vs. A/F R



CMU 94 data

Productivity



1997 SEI Study

PSP Results for Industrial Use

- Although there have been some notable successes [Ferguson 1997], replicating the results in PSP training in an industrial setting has not been easy.
  - In most cases, the classroom PSP has to be adapted/tailored to the development environment and the domain of application.
  - PSP requires a great deal of discipline and most people need a supportive environment to maintain such discipline.
  - PSP does not fit well in an organization with a low maturity level.
- More positive results have been achieved when the PSP is coupled with the Team Software Process[SM] (TSP[SM]) [Davis 2003].

Messages to Remember

- The PSP implements CMM key practices at the individual and team level respectively.
  - The PSP is a “level 5” processe.
- The PSP is flexible and tailorable, but requires discipline and commitment.
- The PSP can
  - reduce schedule deviation
  - improve the quality of the delivered product
  - improve productivity (or cause not decrease)

References

- [Davis 2003] Davis, N. and Mullaney, J., *The Team Software Process (TSP) in Practice: A Summary of Recent Results*, CMU/SEI-2003-TR-014, Software Engineering Institute, Carnegie Mellon University, September 2003.
- [Ferguson 1997]Ferguson, P., Humphrey, W., Khajenoori, S., Macke, S., and Matvya, A. "Introducing the Personal Software Process: Three Industry Case Studies," *Computer*, pp. 24-31, May 1997.
- [Hayes 97] Hayes, W. and Over, J.W., *The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers*, CMU/SEI-97-TR-001, Software Engineering Institute, Carnegie Mellon University, December 1997.
- [Humphrey 1995] Humphrey, Watts S., *A Discipline for Software Engineering*, Addison Wesley, 1995
- [Humphrey 1997] Humphrey, Watts S., *Introduction to the Personal Software Process*, Addison Wesley, 1997.
- [Paulk 1993] Paulk, Mark C., et. al., *Capability Maturity Model for Software*, Version 1.1, CMU/SEI-93-TR-024, Software Engineering Institute, Carnegie Mellon University, 1993.
- [SEMA 2003] Software Engineering Measurement and Analysis, *Process Maturity Profile: Software CMM, CBA IPI and SPA Appraisal Results, 2003 Mid-Year Update*, Software Engineering Institute, Carnegie Mellon University September 2003, http://www.sei.cmu.edu/sema/pdf/SW-CMM/2003sepSwCMM.pdf.