HELSINKI UNIVERSITY OF TECHNOLOGY

Department of Computer Sciece

Laboratory of Information Processing Science

Virpi Kaltio

Implementing a University Course based on the Personal Software Process (PSP)

Master´s Thesis

Supervisor: Professor Reijo Sulonen

Instructor: M.Sc. Kari Alho

HELSINKI UNIVERSITY OF TECHNOLOGY

ABSTRACT OF THE MASTER'S THESIS

Author:	Virpi Hannele Kaltio		
Title:	Implementing a University Course based on the Personal Software Process (PSP)		
Date:	14.08.1998	**Number of pages:**	98
Department:	Department of Civil and Environment Engineering.		
Professorship:	Tik-76 Information Processing Science		
Supervisor:	Professor Reijo Sulonen, Helsinki University of Technology		
Instructor:	M. Sc. Kari Alho, Helsinki University of Technology		

This thesis has two goals. The first goal is to implement a Personal Software Process (PSP) course, and evaluate the suitability of the PSP for the Helsinki University of Technology Computer Science (CS) curriculum. The second goal is to gather improvement ideas for future PSP courses. This study is based on a pilot PSP course during semester 1997-1998.

The theoretical part of thesis covers the architecture of the Personal Software Process (PSP), most used software process improvement approaches (SW-CMM, IDEAL Model, Quality Improvement Paradigm (QIP), and Experience Factory), and the PSP experiences and results from different universities and industry.

The applied part of the thesis deals with the pilot PSP course at Helsinki University of Technology, and its experiences and critical improvement ideas for future PSP courses.

Keywords: personal software process, process development, process definition, process improvement, software engineering education, software process training

TEKNILLINEN KORKEAKOULU — DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Virpi Hannele Kaltio		
Työn nimi:	Personal Software Process (PSP) menetelmään perustuvan korkeakoulukurssin toteuttaminen		
Päivämäärä:	14.08.1998	**Sivumäärä:**	98
Osasto:	Rakennus- ja yhdyskuntatekniikan osasto.		
Professuuri :	Tik-76 Tietojenkäsittelyoppi		
Työn valvoja:	Professori Reijo Sulonen, Teknillinen korkeakoulu		
Työn ohjaaja:	DI Kari Alho, Teknillinen korkeakoulu		

Tällä diplomityöllä on kaksi tavoitetta. Ensimmäinen tavoite on toteuttaa Personal Software Process (PSP)-kurssi, ja arvioida sen soveltuvuutta osaksi Helsingin teknillisen korkeakoulun tietotekniikan koulutusohjelmaa. Toinen tavoite on kerätä parannusehdotuksia tulevien PSP-kurssien kehittämiseksi. Diplomityö perustuu lukuvuonna 1997-1998 järjestettyyn pilotti PSP-kurssiin.

Diplomityön teoreettisessa osassa esitetään Personal Software Process (PSP) arkkitehtuuri, tunnetuimmat ohjelmistoprosessin kehittämismallit (SW-CMM, IDEAL-malli, Quality Improvement Paradigm (QIP) ja Experience Factory) sekä eri yliopistojen ja teollisuuden kokemuksia ja tutkimustuloksia PSP:n käytöstä.

Diplomityön soveltava osa käsittelee Teknillisessä korkeakoulussa järjestettyä PSP-kurssia, siitä saatuja kokemuksia sekä kriittisiä parannusehdotuksia tulevien PSP-kurssien kehittämiseksi.

Avainsanat: henkilökohtainen ohjelmistoprosessi, prosessin kehittäminen, prosessin määrittely, prosessin parannus, ohjelmistotuotannon opetus, ohjelmistoprosessi koulutus

ACKNOWLEDGEMENTS

This thesis was written at Helsinki University of Technology at the Laboratory of Information Processing Science during the semester 1997-1998.

I would like to thank my supervisior Professor Reijo Sulonen and my instructor Kari Alho for their comments on the thesis.

I would like to thank all these students, who have participated at the pilot PSP course. Your experiences and comments have been very valuable for me and my thesis.

Special thanks to my sweetheart husband and my best friend Timo Kaltio for encouraging me to finish my studies and this thesis. You have been extremely helpful in the definition of both structure and the contents of the thesis.

Otaniemi, 14.08.1998

Virpi Kaltio

TABLE OF CONTENTS

TERMINOLOGY

In this section the principal process terms are presented according to IEEE standards 610.12-1990 Standard Glossary of Software Engineering Terminology [IEEE90], Software Process Development and Enactment: Concepts and Definitions [FeHu93], The Capability Maturity Model - Guidelines for Improving the Software Process [Paul94], and Personal Software Process (PSP) - Process Terms [Proc98].

Activity	Any step taken of or function performed, both mental and physical toward achieving some objective. Activities include all the work the managers and technical staff do to perform the tasks of the project and organization. [Paul94]
A/FR	A/FR refers to the ratio of the appraisal COQ to the failure COQ. [Proc98]
Checklist	A checklist is a list of steps engineers use to ensure that all parts of a process are followed. The Personal Software Process (PSP) uses design review and code review checklists. [Proc98]
COQ	The cost of quality is the percentage of development time spent on quality related activities. Appraisal COQ refers to the time appraising product quality. In the PSP, appraisal COQ includes on the design and code review times. Failure COQ refers to the time spent repairing defects. In the PSP, failure COQ includes only the compile and test times. [Proc98]

CMM	The Capability Maturity Model is a framework for describing and evaluating the capability of software organizations. [Proc98]
CMU	Carnegie Mellon University
CS	Computer Science
Defect	A defect is anything that is wrong with the program that must be fixed before the program can be developed, tested, enhanced, or used. The PSP counts all defects found in reviews, compiling, or testing. [Proc98]
Defect Type	Defects are classified by types according to their characteristics. An analysis of defect types helps engineers determine which defect categories cause them the most trouble and how to better find or prevent them. [Proc98]
ERAU	Embry Riddle Aeronautical University
Estimating Error	The percentage error of an estimate is 100*(actual-estimate)/(estimate). [Proc98]
Fix Time	The fix time is the elapsed working time from when an engineer first determined there was a defect until the defect was fixed and the fix verified. [Proc98]
HUT	Helsinki University of Technology

IDEAL	The SEI approach to the cycle of software process improvement, based on intiating an improvement effort, diagnosing the software process, establishing mechanisms for improving the process, acting to implement the improvements, and leveraging them across the organization. [Paul94]
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Organization for Standardization
Injected Defect	An engineer's mistake often produces an incorrect element of the design or an incorrect program statement. This is called injecting a defect. The phase injected refers to the process phase in which the engineer was working when the defect was injected. [Proc98]
Key practice (KP)	The infrastructure and activities that contribute most to the effective implementation and institutionalization of a key process area. [Paul94]
Keyprocess area (KPA)	A cluster of related activities that, when performed collectively, achieve a set of goals considered to be important for establishing process capability. The key process areas have been defined to reside at a single maturity level. They are the areas identified by the SEI to be the principal building blocks to help determine the software process capability or an organization and understand the improvements needed to advance to higher maturity levels. [Paul94]

KLOC	Thousands of LOC. [Proc98]
Maturity level	A well-defined evolutionary plateau toward achieving a mature software process. The five maturity levels in the SEI´s Capability Maturity Model are initial, repeatable, defined, managed and optimizing. [Paul94]
LOC	LOC stands for lines of code. This is a measure of program size that counts every text line that contains a programming instruction. [Proc98]
Module	A relatively small element of a programming project that is implemented by an individual engineer. Module sizes typically range from 50 to 5,000 LOC. [Proc98]
Phase	A process phase is a defined part of the process, such as design or test. [Proc98]
PIP	Process Improvement Paradigm
PROBE	PROBE stands for PROxy Based Estimating, a size and development time estimating method used in the PSP. [Proc98]
Process	A set of partially ordered steps intended to reach a goal. [FeHu93]
Process capability	The range of expected results that can be achieved by following a process. [Paul94]

Process definiton	An implementation of a process design in the form of a partially ordered set of process steps that is enactable. [FeHu93]
Process development	The act of creating enactable processes. It may include planning, architecture, design, instantiation, and validation. [FeHu93]
Product	See *software product and software work product.*
Productivity	While there are many measures of programming productivity, the PSP uses the number of LOC developed per hour. [Proc98]
Project	An undertaking requiring concerted effort that is focused on developing and/or maintaining a specific product. The product may include hardware, software, and other components. Typically a project has its own funding, cost accounting, and delivery schedule. [Paul94]
Proxy	A proxy is a substitute. In the PSP, proxies are used to help engineers visualize the sizes of their products when they make size estimates. The principal proxies used in PSP are objects and procedures. Some of the other possible proxies are function points, screen, and document chapters. [Proc98]
PSP	The Personal Software Process is a family of defined processes that guide engineers in measuring, evaluating, and improving the way they do their software engineering tasks. [Proc98]

Removed Defect	All defects that are found and fixed are counted as removed defects by the PSP. The phase when the defect was repaired is the phase removed. [Proc98]
Review	A review is a personal check of the program made by the engineer who developed it. Design and code reviews are used in the PSP to find and fix defects as early in the process as possible. [Proc98]
SEI	Software Engineering Institute
SW	Software
Quality	(1) The degree to which a system, component, or process meets specified requirements. (2) The degree to which a system, component, or process meets customer or user needs or expectations. [IEEE-STD-610]
Size	In the PSP, the size of software products is measured in LOC. [Proc98]
Software process	A set of activities, methods, practices, and transformations to develop and maintain software and the associated products (i.e., project plans, design documents, code, test cases, and user manuals). [Paul94]
Software product	The complete set, or any of the individual items of the set, of computer programs, procedure, and associated documentation and data designated for delivery to customer or end user. [IEEE-STD-610]

Software project	An undertaking requiring concerted effort and focusing on analyzing, specifying, designing, developing, testing, and/or maintaining the software components and associated documentation of a system. A software project may be part of a project building a hardware/software system. [Paul94]
Software work product.	Any artifact created as part of defining, maintaining, or using a software process. They can include process descriptions, plans, procedures, computer programs, and associated documentation, which may or may not be intended for delivery to a customer or end user. [Paul94]
Task	(1) A sequence of instructors treated as a basic unit of work. [IEEE-STD-610] (2) A well-defined unit of work in the software process that provides management with a visible checkpoint into the status of the project. Tasks have readiness criteria (preconditions) and completion criteria (postconditions). [Paul94]
TAME	Tailoring A Measurement Environment
TSP	Team Software Process
TSPe	Team Software Process for Education
Template	A template is a variable length form. Templates are needed when the kinds of data to be entered are well known but the amount of data is variable. [Proc98]

Yield	Yield is the percentage of defects removed from a software product during a given process phase. Process yield is the percentage of defects injected in the product before the first compile that are removed before the first compile. [Proc98]

1. INTRODUCTION

1.1 Background of The Thesis

The work presented in this thesi was performed during the semester 1997-1998, on a pilot Personal Software Process (PSP) course at Helsinki University of Technology (HUT). The idea of implementing the PSP standard course, and evaluating its suitability for HUT Computer Science curriculum originated from Professor Reijo Sulonen´s at Laboratory of Information Processing Science, HUT. 16-19th June 1997 I took part in European Software Engineering Process Group Conference Amsterdam where Mr. Watts S. Humphrey gave very interesting the PSP tutorial. Both professor Reijo Sulonen and I were interested in the same professional topic, and August 1997 we decided to start the pilot project.

The pilot PSP course was carried out with co-operation with the graduate level Tik-76.115 Software Project course. The pilot PSP group began with an initial enrollment of twelve students, nine of which took the course final. The course was taught using *Introduction to the Personal Software Process* as the text book. We had problems to obtain these study books and Humphrey´s *Instructors Guide and Assignment Kits* from Addison-Wesley, and we had not much time to study the PSP method. Because of these problems we had to find an effective way to teach the PSP method. The students were separeted into two PSP seminar groups. Each seminar group met regularly two or three hours once a week during October. At the beginning of each seminar session each student gives a summary of a text book chapter, after which we discuss the topic. This way the students took in the PSP principles very easy, and they had the ability to start using the PSP methods and forms from the beginning of their software project work. Most student seminar lectures were done very well, and it helped to commit to PSP.

All students had to do their own PSP work, but they did not have to work alone. Students had to make their own estimates, do their own designs, code their own programs, and compile and test their own work. After their software project work I gathered information with feedback questionnaire from their subjective PSP course experiences, and collected ideas for the future improvement of the PSP course.

1.2 Objectives of the Thesis

This thesis had two goals. The first goal was to implement the Personal Software Process (PSP) course, and to evaluate the suitability of PSP for the Computer Science (CS) curriculum at Helsinki University of Technology. The second goal was to gather improvement ideas for future PSP courses. The purpose of this thesis was not study any PSP data, for example, defects per KLOC, Yields, A/FR. These measurements will be described shortly, but the main purpose of this thesis is to gather relevant information from student's subjective PSP experiences. Software Engineering Institute (SEI) has published a lot of metrics data.

1.3 Structure of the Thesis

The remainder of this thesis is structured as follows:

Chapter 2, *The Personal Software Process (PSP)*, describes the architecture of the Personal Software Process (PSP). First, a general overview is given. Then, each component of the architecture is discussed in greater detail.

Chapter 3, *The Organizational Software Process Improvement Approaches,* discusses most commonly used software process improvement approaches, that have been published in the software management and engineering literature.

Chapter 4, *Reported Experiences and Results of Teaching and Using the Personal Software Process (PSP),* contains PSP experiences and results from different universities and industry.

Chapter 5, *The Pilot PSP Course at Helsinki University of Technology,* describes how the pilot PSP course was organized at Helsinki University of Technology (HUT).

Chapter 6, *Analysis of the Results from the Pilot PSP course,* reports on student's experiences of the pilot PSP course at Helsinki University of Technology, and introduces some critical improvement ideas for the future PSP courses.

Chapter 7, *Conclusions,* presents the main findings of the thesis.

2. THE PERSONAL SOFTWARE PROCESS

2.1 Overview of the Personal Software Process (PSP)

The Personal Software Process (PSP) is a promising approach to help individual software developers improve the predictability, quality, and productivity of their work. The PSP is a defined and measured software process designed to be used by individual software engineers. The PSP was developed by Watts Humphrey of the US Software Engineering Institute (SEI) and is described in his book *A Discipline for Software Engineering* [Hump95a]. The PSP has been developed to address the need for process improvement in small organizations and small project teams. It aims to show individuals how to define and apply personal working methods incorporating best practice, and then to adjust them in the light of analysis to optimize their performance. The research work on the PSP started in 1989.

The original impetus for developing the PSP came from questions about the Software Engineering Institute's (SEI) Capability Maturity Model (CMM) [Hump94d]. Like the CMM, the PSP is based on process improvement principles. 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 [Hump94b]. While the CMM is focused on improving organizational capability, the focus of the PSP is the individual engineer. To foster improvement at the personal level, PSP extends process management and control to the software engineer. With PSP, engineers develop software using a disciplined, structured approach.

2.2 The Personal Software Process (PSP) Principles

Many software engineers spend the bulk of their time solving problems that are just like those they have solved many times before. In fact software organizations have been solving these same or similar problems for many years. After software engineers spend years solving old and familiar problems, many of them feel that they have wasted their time. However many engineers worry that the use of a disciplined process will constrain their creativity and productivity. The initial PSP data show that very few engineers use such proven practices as disciplined design methods, design or code reviews, or defined testing procedures [Hump94b].

The PSP approach is based on the following principles: [Hump94c]

- "By defining, measuring, and tracking their work, software professionals will better understand what they do.
- This understanding will enable the engineers to better recognize what methods work best for them and see how they can more consistently apply them.
- The engineers will then have a defined process structure and measurable criteria for evaluating and learning from their own and others' experiences.
- With this knowledge, the engineers can select those methods and practices that best suit their particular tasks and abilities.
- By using a customized set of orderly, consistently practiced, and high quality personal practices, the engineers will be more effective members of their development teams and projects."

2.3 The Personal Software Process (PSP) Strategy

The Capability Maturity Model for Software (SW-CMM) is an organizational-focused process improvement framework [Hump89, Paul94]. The SW-CMM will be described in chapter 3.1.1. While the CMM enables and facilitates good work, it does not guarantee it. The engineers must also use effective personal practices. The CMM provides a body of software engineering practices that have been found effective for large-scale software development [PaulCC93]. Starting with the CMM, Watts Humphrey selected and defined a subset of the CMM key process areas for use by individual software practitioners [Hump94c].

Some CMM items are not included in the PSP because their effectiveness cannot be demonstrated at the individual level, for example software subcontract management and intergroup coordination. Some items, for example requirements management and software configuration management, these can be usefully practiced by individuals but their implications are better demonstrated in a small-team environment [Hump95a]. The CMM key process areas (KPAs) are shown in Figure 2.1 and those that are at least partially addressed by the PSP are shown in italics and noted with an asterisk.

The PSP demonstrates the goal of 12 of the 18 CMM key process areas. The figure below shows which of the KPAs of the CMM are demonstrated by PSP.

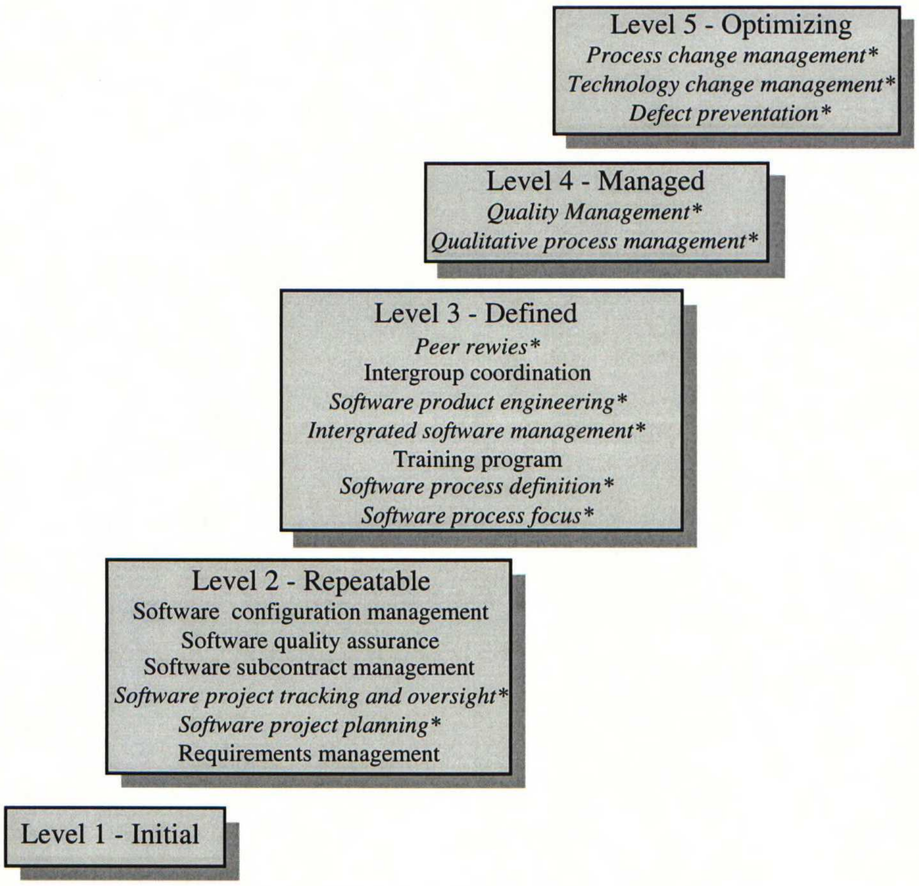

Level 5 - Optimizing
*Process change management**
*Technology change management**
*Defect preventation**
Level 4 - Managed
*Quality Management**
*Qualitative process management**
Level 3 - Defined
*Peer rewies**
Intergroup coordination
*Software product engineering**
*Intergrated software management**
Training program
*Software process definition**
*Software process focus**
Level 2 - Repeatable
Software configuration management
Software quality assurance
Software subcontract management
*Software project tracking and oversight**
*Software project planning**
Requirements management
Level 1 - Initial

Figure 2.1. The CMM and the PSP [Hump95a].

2.4 The Personal Software Process (PSP) Evolution

The PSP has a maturity structure much like the CMM. The PSP is introduced using seven progressive process steps. Each PSP step, shown in Figure 2.2, includes all the elements of prior steps together with one or two additions. This minimizes the impact of process change on the engineer, who needs only to adapt the new techniques into an existing baseline of practices [HaOv97].

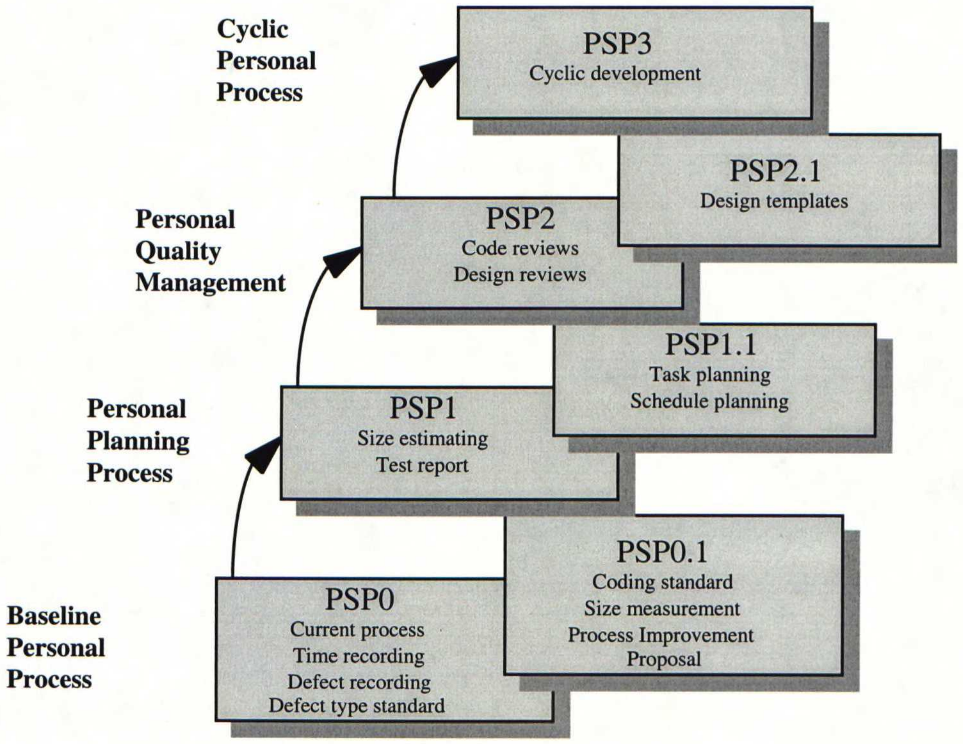


Figure 2.2. The PSP Evolution [Hump95a].

The PSP takes the practitioners through a set of evolutionary stages called the Baseline Process (PSP0), the Personal Planning Process (PSP1), the Personal Quality Management (PSP2), the Cyclic Personal Process (PSP3) and the Team Software Process (TSP). The PSP progression is described in the following paragraphs.

2.4.1 PSP0 - The Baseline Personal Process

In the PSP, the first step is to establish a baseline that includes measurements and a reporting format. In this step, engineers use their current design and development methods. With PSP0, they learn to how to apply the PSP forms and scripts to their personal work. They do this by measuring the time spent by phase and the defects found in compile and test. This lets engineers gather real, practical data on their personal processes and gives them benchmarks against which they measure progress while learning and practicing the PSP.

PSP is enhanced to PSP0.1 by adding a coding standard, size measurement, and the Process Improvement Proposal (PIP) form. The PIP is a form that provides a structured way to record process problems, experiences and improvement suggestions [Hump95a].

2.4.2 PSP1 - The Personal Planning Process

PSP1 adds planning steps to PSP0. The initial step adds size and resource estimation and a test report. In PSP1, the Proxy Based Estimation (PROBE) method is introduced. With PSP1, engineers used PROBE to estimate the sizes and development times of the products they produce. They use their historical data to improve their estimate. The PROBE estimating method is shown in Figure 2.3.

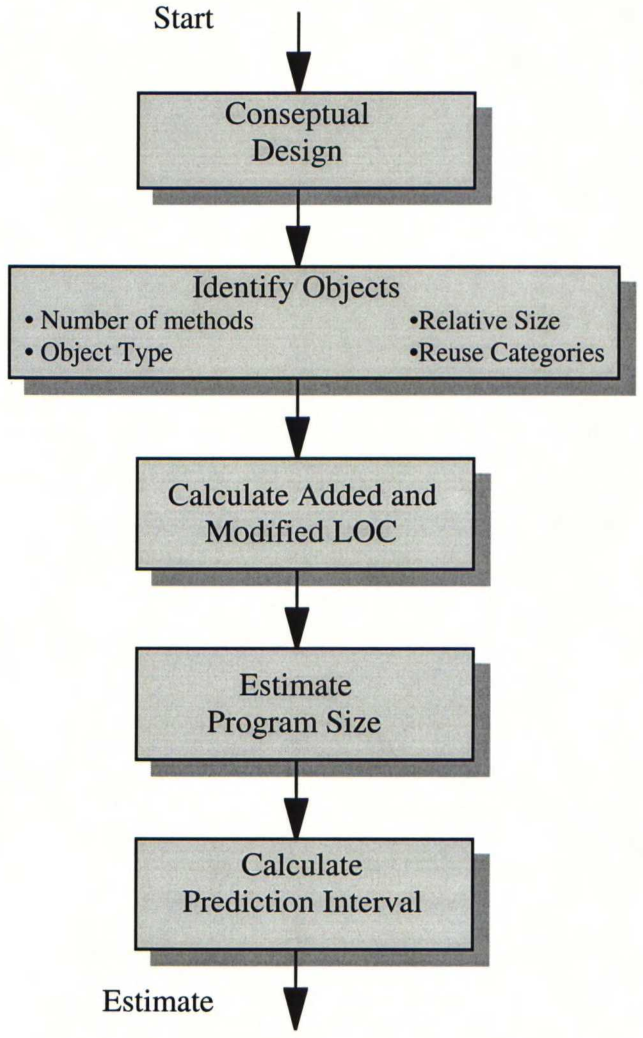


Figure 2.3 The PROBE Estimating Method [Hump97c].

PROBE is a PROxy-Based Estimating method in which engineers use data on their previously developed programs to estimate the size and development time for a new program. PROBE uses statistical techniques to calculate the most likely size and time estimates and the likely range of these estimates [Hump96b].

Schedule and task planning are added in PSP1.1. Engineers want or need to make explicit, documented plans for their work for the following reasons: [Hump95a]

- "Understand the relation between the size of the programs they develop and the times they take to develop them,
- Learn how to make commitment that they can meet,
- Prepare an orderly plan for doing their work, and
- Establish a basis for tracking their work."

2.4.3 PSP2 - The Personal Quality Management

One PSP goal is to help engineers manage defects they inject. PSP2 adds personal design and code reviews to PSP1. These help the engineers to find defects earlier in their processes and to appreciate the benefits of finding defects early.

PSP2.1 introduces design specification and analysis techniques. The intent is not to tell engineers how to do design but to address the criteria for design completion. PSP2.1 establishes design completeness criteria and examines various design verification and consistency techniques [Hump95a].

2.4.4 PSP3 - A Cyclic Personal Process

PSP3 is the final PSP step. The earlier PSP processes concentrate on the linear process for building small programs. In the PSP3, engineers scale up PSP methods to larger projects. With PSP3, engineers see how to use the PSP for large scale work such as cyclic development and issue tracking log. They learn how to adjust their personal processes for different types of work. The Figure 2.4 illustrates how engineers can couple multiple PSP2.1 processes in a cyclic fashion to scale up to developing modules with as many as several thousands LOC.

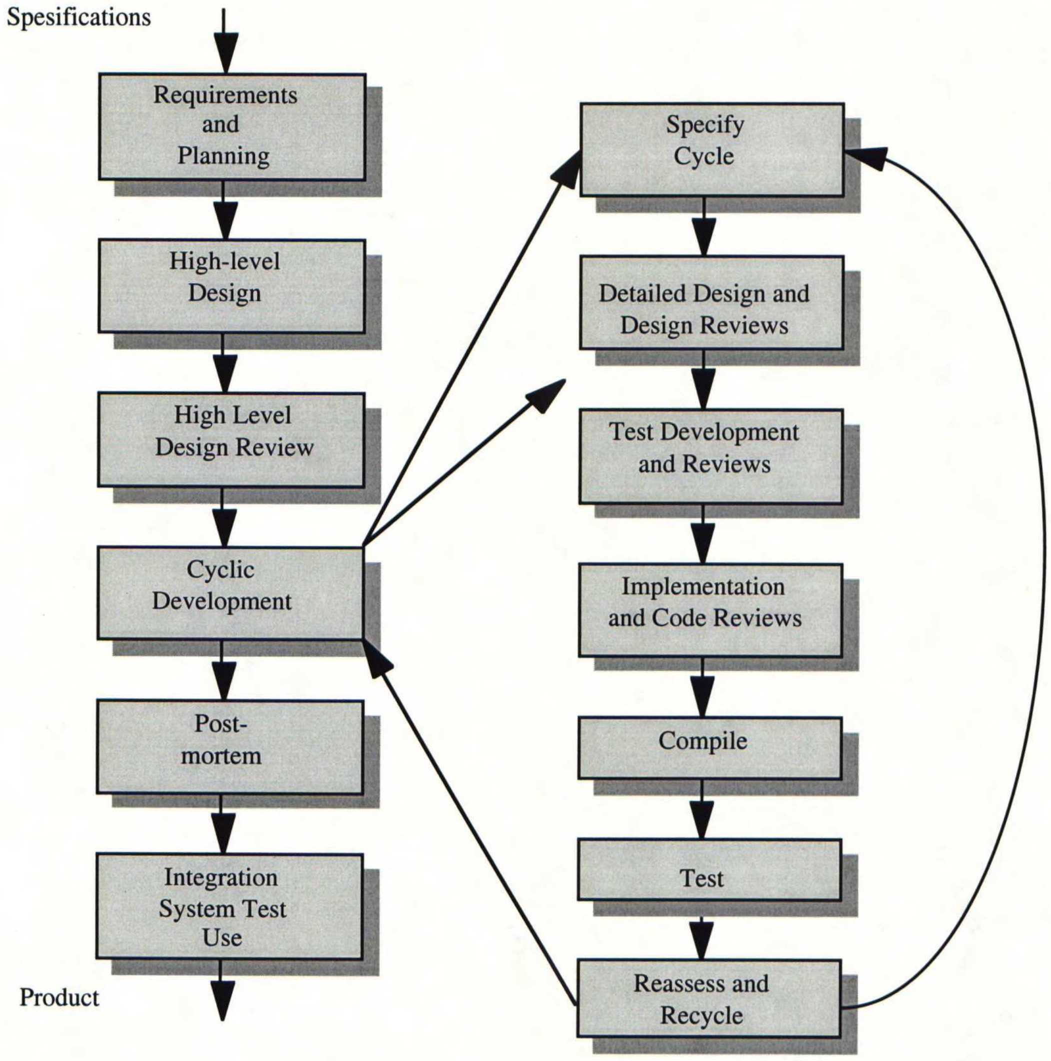

Spesifications
Requirements and Planning
High-level Design
High Level Design Review
Cyclic Development
Post-mortem
Integration System Test Use
Product
Specify Cycle
Detailed Design and Design Reviews
Test Development and Reviews
Implementation and Code Reviews
Compile
Test
Reassess and Recycle

Figure 2.4 The PSP3 Process - The Cyclic Development Process [Hump94c].

"The first build is a base module or kernel that is enhanced in iterative cycles. In each iteration, a complete PSP2 is used, including design, code, compile, and test. Since each builds on the previously completed increments, the PSP3 process is suitable for programs of up to several KLOC." [Hump94c]

2.4.5 TSP - The Team Software Process

Most software is developed by groups. Software development is generally thought as a solo activity but when engineers go into industry, most of them work on teams. The Team Software Process's (TSP) principal objective is to show PSP-trained engineers how to run a team-based project.

The Team Software Process has five objectives: [Hump98c]

- "Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure software teams or integrated products teams (IPT) of three to about 20 engineers.
- Show managers how to coach and motivate their teams and how to help them sustain peak performance.
- Accelerate software process improvement by making CMM Level 5 behavior normal and expected.
- Provide improvement guidance to high-maturity organizations.
- Facilitate university teaching of industrial-grade team skills."

Team Software Process (TSP) is still in development and is not generally available, except through SEI-supported PSP transition collaborations. The SEI is working with a few universities and a number of industrial organizations on introduction the TSP and the early results are encouraging [Hump98d]. Several organizations are currently using the PSP, such as Adtranz, Baan, Boeing, Embry-Riddle Aeronautical University, and Teradyne [Hump98e].

2.5 The Personal Software Process (PSP) and The Team Software Process versus SW-CMM

The Capability Maturity Model (CMM) for software, the Personal Software Process (PSP), and the Team Software Process (TSP) are methods of software process improvement that were developed at the Software Engineering Institute (SEI). "The CMM provides an overall framework to describe the activities software organizations need to consistently produce effective results; the PSP helps engineers use process principles in their personal work and the TSP shows integrated product teams how to use these processes to consistently produce quality products on aggressive schedules and for their planned costs" [Hump98d].

The CMM, PSP and TSP provide an integrated three-dimension framework for process improvement. The Table 2.1 below shows which of the KPAs of the CMM are demonstrated by PSP and TSP. The CMM has 18 key process areas, and the PSP (12 KPAs) and the TSP (16 KPAs) guide engineers in addressing almost all of them.

Level	Focus	Key Process Areas	PSP	TSP
5 Optimizing	Continuous Process Improvement	Defect Prevention	x	x
		Technology Change Management	x	x
		Process Change Management	x	x
4 Managed	Product and Process Quality	Quantitative Process Management	x	x
		Software Quality Management	x	x
3 Defined	Engineering Process	Organization Process Focus	x	x
		Organization Process Definition	x	x
		Training Program		
		Integrated Software Management	x	x
		Software Product Engineering	x	x
		Intergroup Coordination		x
		Peer Reviews	x	x
2 Repeatable	Project Management	Requirements Management		x
		Software Project Planning	x	x
		Software Project Tracking	x	x
		Software Quality Assurance		x
		Software Configuration Management		x
		Software Subcontract Management		

Table 2.1. PSP and TSP coverage of CMM key process areas [Hump98c].

2.6 The PSP Course Structure and Assignments

The SEI has developed two university-level PSP courses: an intensive one-semester graduate or senior-level undergraduate course, and a course for freshmen [Hump95a, Hump97b]. The objectives, prerequisites, structure, and support for these courses are shown in Appendix 1 and 2. The SEI has designed a Team Software Process for Education (TSPe) course based on early experiences with TSP teams. This course is planned for either one- or two-semester dedicated courses or for shorter projects in other courses. The TSPe course objective, prerequisites, structure, and support are summarized in Appendix 3.

The PSP textbook, *A Discipline for Software Engineering* [Hump95a], describes a standard PSP course structure. It describes a one-semester curriculum for advanced undergraduates or graduate students in computer science that teaches concepts in empirically guided software process improvement. The PSP is structured as courses where the software engineer must follow the proposed sequence of exercises, fill in tables, and identify their critical points through statistical calculations on data collected during the development of the programming exercises. The PSP course structure, shown in Table 2.2, includes the course topics covered in the lecture and assigned reading, the associated programming exercise, its description, and the PSP process level used for the assignment. The PSP contains 19 exercise programs (1A to 10A and 1B to 9B) and five report assignments (R1 to R5). The ten A series of programs and the five report assignment form the basic set used while learning the PSP.

Course Topic	Exercise	Exercise Description	PSP Level
The Personal Software Process Strategy and the Baseline PSP	1A	Calculate the mean and standard deviation of N real numbers stored in a linked list	PSP0
The Planning Process	2A R1 R2	Count the LOC in a program source file Produce a LOC counting standard: Count logical LOC in the language you use to develop the PSP exercises Produce a coding standard	PSP0.1
Measuring Software Size	3A	Enhance program 2A to count object or function/procedure LOC	PSP0.1
	R3	Defect analysis report: Analyze the defects for programs 1A to 3A	
Estimating Software Size	4A	Calculate the linear regression parameters for N pairs of real numbers stored in a linked list	PSP1
Resource and Schedule Estimating	5A	Numerical integration using Simpson´s rule	PSP1.1
Measurements in the Personal Software Process	6A	Enhance program 4A to calculate a 90% and 70% prediction interval	PSP1.1
Design and Code Reviews	R4	Midterm process analysis report	
Software Quality Management	7A	Calculate the correlation of N pairs of real numbers stored in a linked list	PSP2
Software Design	8A	Sort a linked list	PSP2 or PSP2.1
Software Design Verification	9A	Chi-square test for normality	PSP2.1
Scaling-up the PSP	10A	Calculate the multiple linear regression parameters for N sets of four real numbers stored in a linked list	PSP3
Defining the Software Processes and Using the PSP	R5	Final process analysis report: The report of process progress, quality, and lessons learned	

Table 2.2. Standard PSP course structure [Hump95a, Hump96a, HaOv97].

3. THE PERSONAL SOFTWARE PROCESS VERSUS ORGANIZATIONAL SOFTWARE PROCESS IMPROVEMENT

3.1 Overview of Software Process Improvement Approaches

3.1.1 The SW Capability Maturity Model (SW-CMM)

The Capability Maturity Model for Software (CMM or SW-CMM) developed by the Software Engineering Institute (SEI) has had a major influence on software process and quality improvement around the world [Paul94]. Continuous process improvement is based on many small, evolutionary steps rather than revolutionary innovations [Paul96]. The Capability Maturity Model for Software is a framework that describes the key elements of an effective software process. The SW-CMM defines an evolutionary path that is to followed by all organizations seeking software process improvement.

There are five possible levels of maturity for a software process in the CMM model. Each level has a number of key process areas (KPAs) that represent the primary issues the organization needs to address in order to mature its process [HeGo96]. The five levels, and the 18 key process areas that describe them in detail, are summarized in Figure 3.1.

The five levels can be briefly described as: [PaulCC93]

1. *Initial*	The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.
2. *Repeatable*	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.

3. *Defined* 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.

4. *Managed* Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

5. *Optimizing* Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

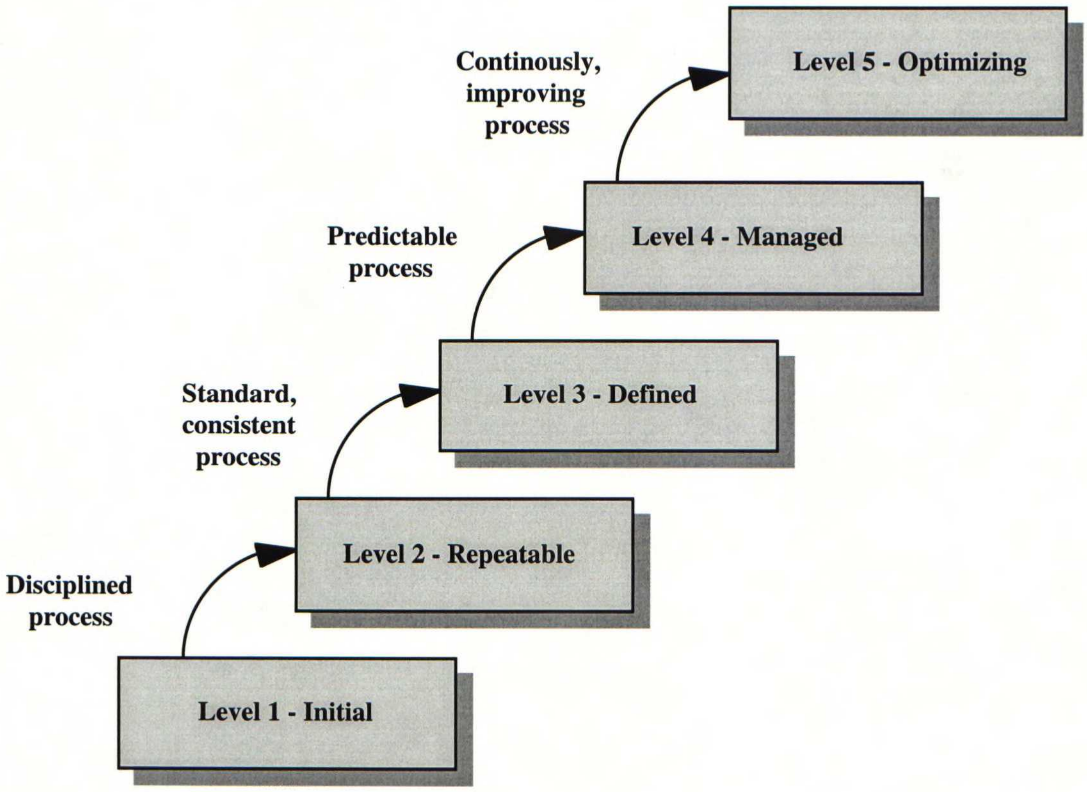


Figure 3.1. The Five Levels of Software Process Maturity [Paul94].

Except for Level 1, each maturity level is decomposed into several key process areas that indicate the areas an organization should focus on to improve its software process [Paul95]. The structure of the CMM is illustrated below in Figure 3.2 and Figure 3.3 lists the key process areas for each maturity level in the CMM. Each maturity level is composed of a set of key process areas (KPAs), and each KPA consists of numerous key practices that accomplish the goal of the process area [PaulCC93]. The *key practices* describe the activities and infrastructure that contribute most to the effective implementation and institutionalization of the key process area. The key practices describe "what" is to be done, but they should not be interpreted as mandating "how" the process should be implemented.

Key practices are organized by common features. The common features are attributes that indicate whether the implementation and institutionalization of a key process area are effective, repeatable, and lasting [PaulCC93].

The five common features are listed briefly:

- Commitment to Perform,
- Ability to Perform,
- Activities Performed,
- Measurement and Analysis, and
- Verifying Implementation.

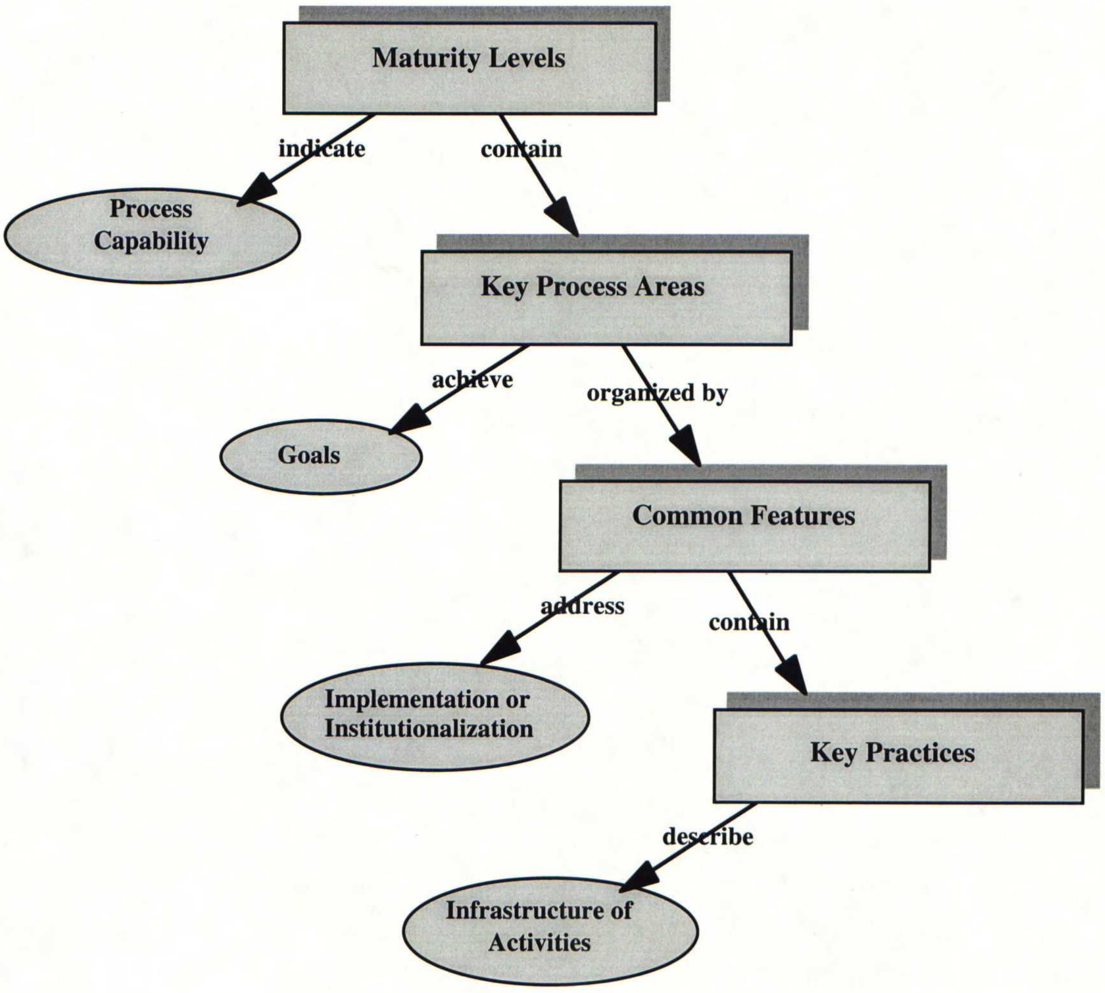
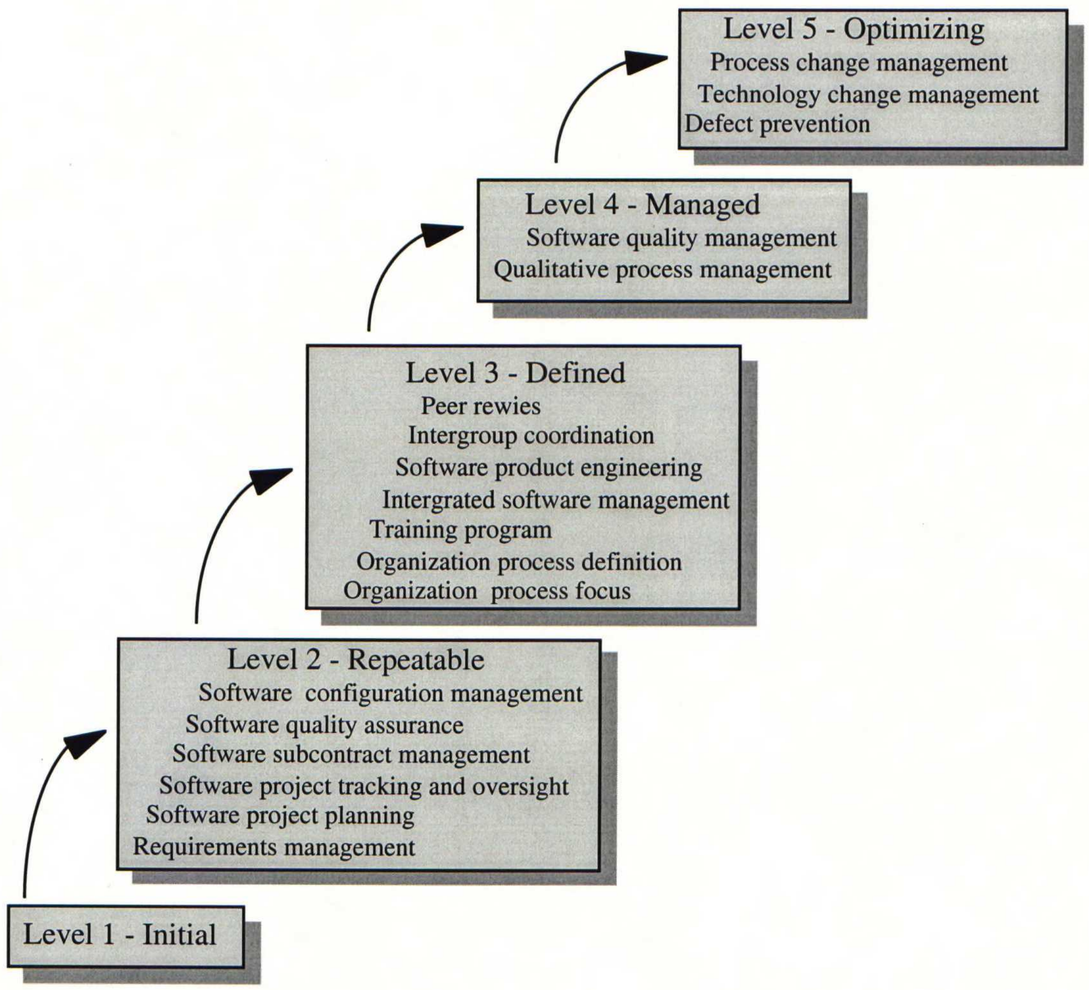


Figure 3.2. The Structure of the Capability Maturity Model [PaulCC93].



Level 5 - Optimizing
Process change management
Technology change management
Defect prevention
Level 4 - Managed
Software quality management
Qualitative process management
Level 3 - Defined
Peer rewies
Intergroup coordination
Software product engineering
Intergrated software management
Training program
Organization process definition
Organization process focus
Level 2 - Repeatable
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Level 1 - Initial

Figure 3.3. The Key Process Areas in the SW-CMM [PaulCC93].

3.1.2 The IDEAL Model

The SEI has developed an organizational and strategic level model for software process improvement (SPI) initiatives, called IDEAL [McFe96]. The IDEAL model is illustrated in Figure 3.4. The name IDEAL is an acronym for it's five phases, illustrated in the figure: Initiating, Diagnosing, Establishing, Acting, and Leveraging. These are briefly described in the figure, and are discussed in detail in *IDEAL: A User's Guide for Software Process Improvement*.

The IDEAL´s key steps can be briefly described as: [Paul94]

1. *Initiating*	The first phase, when sponsorship and the software process improvement infrastructure are defined and established.
2. *Diagnosing*	The second phase, when appraisals are conducted to establish the software process maturity baseline of the organization and a set of recommendations for improvement are communicated to the organization.
3. *Establishing*	The third phase, when a software process improvement infrastructure is built, including the formation of process action teams and the definition of software process improvement strategic and tactical plans.
4. *Acting*	The fourth phase, when the improvements are implemented.
5. *Leveraging*	The final phase, when lessons learned from the software process improvement effort are analyzed, resulting in updates to the software process improvement process. Sponsorship is renewed and new goals are set for the next improvement cycle.

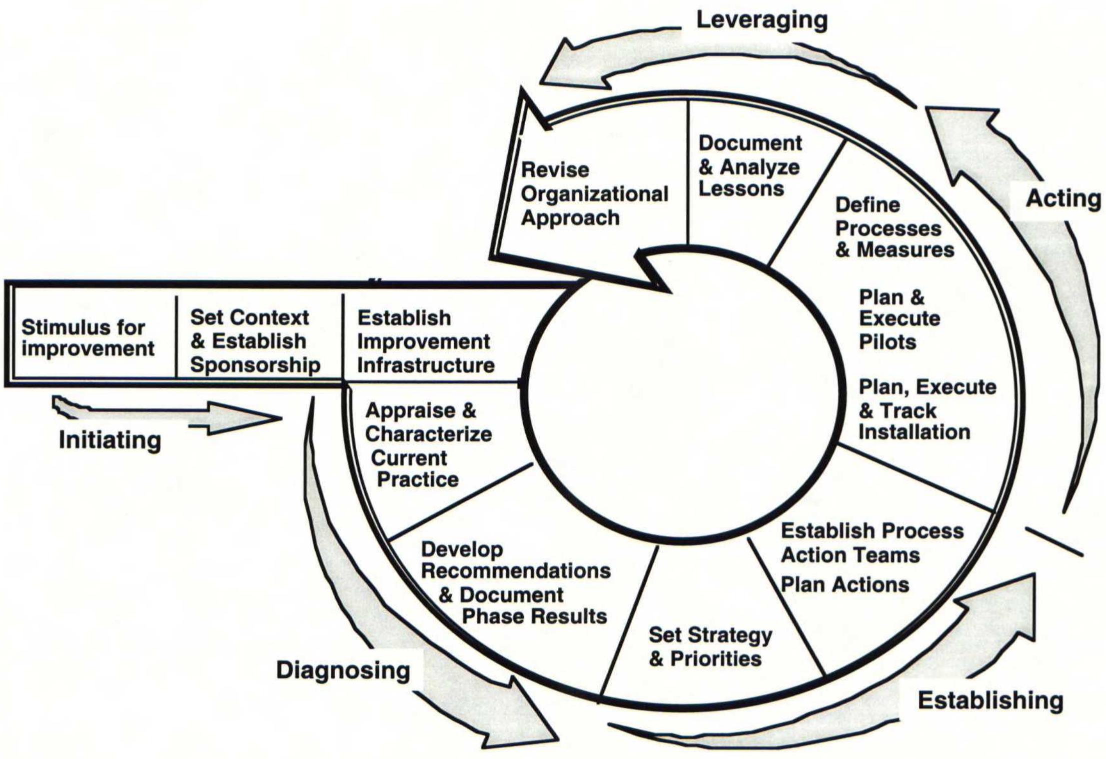

Leveraging
Revise Organizational Approach
Document & Analyze Lessons
Define Processes & Measures
Acting
Plan & Execute Pilots
Plan, Execute & Track Installation
Stimulus for improvement
Set Context & Establish Sponsorship
Establish Improvement Infrastructure
Initiating
Appraise & Characterize Current Practice
Develop Recommendations & Document Phase Results
Establish Process Action Teams
Plan Actions
Set Strategy & Priorities
Diagnosing
Establishing

Figure 3.4. The IDEAL Model [McFe96].

3.1.3 The Quality Improvement Paradigm (QIP)

Quality Improvement Paradigm developed by Victor R. Basili, et al., is the result of the application of the scientific method to the problem of software quality improvement. The QIP is a consept for learning and improvement.

The QIP is articulated into following six steps: [BaRo88, BaMc95]

1. *Characterize*	Characterize the current project and its environment with respect to models and metrics.
2. *Set Goals*	Set quantifiable goals for successful project performance and improvement.

3. *Choose Process*	On the basic of the characterization of the environment and of the goals that have been set, choose the appropriate process model and supporting methods and tools for this project.
4. *Execute*	Execute the processes, construct the products, collect and validate the prescribed data. Perform the processes constructing the products and providing project feedback based upon the data on goal achievement that are being collected.
5. *Analyze*	At the end of each specific project, analyze the data to evaluate the current practices, determine problems, record findings, and recommend future project improvements.
6. *Package*	Package the experience in the form of models and other forms of structured knowledge, and store it in an experience base to be reused on future projects.

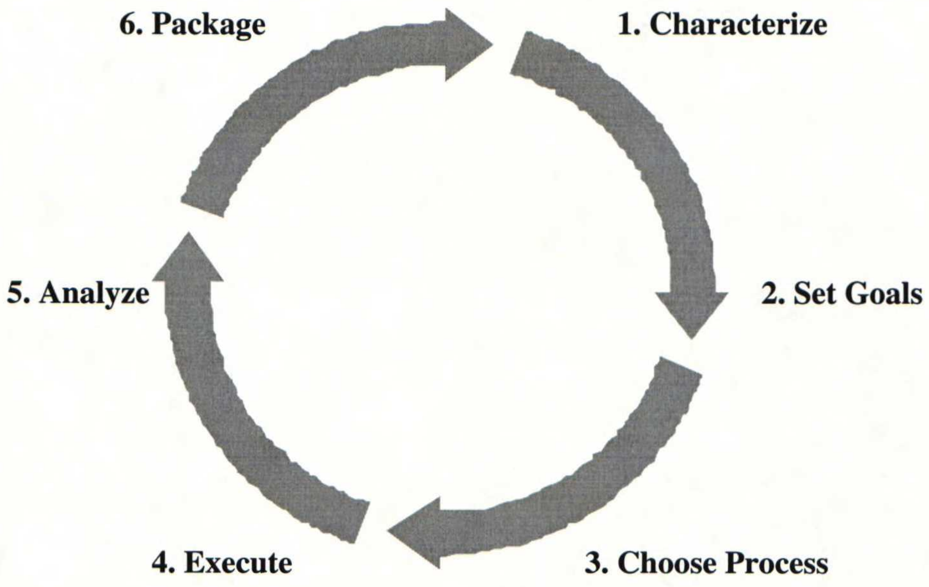


Figure 3.5. The Steps of the Quality Improvement Paradigm (QIP) [BaCR94].

3.1.4 The Experience Factory

The Experience Factory is a logical, organizational and operational model for continuous process improvement in a project-oriented organization. It was developed by professor Victor Basili at NASA/Goddar Flight Center SFC and Software Engineering Laboratory (SEL) at University of Maryland. "The Experience Factory packages experience by building informal, formal or schematized, and productized models and measures of various software process, products, and other forms of knowledge via people, documents, and automated support [BaCR94b]."

The aim of the Experience Factory model is to form a continuously learning organization by systematically collecting, assessing and distributing project experiences by using measurements to validate process improvement efforts.

The focus and priority of the Experience Factory is to support project development: [BaMc95]

- Analyzing experience drawn from people, documents, and automated tools.
- Synthesizing that experience into process models and measures both informal and formal.
- Store these models and schematics in a repository.
- Supply the experience to various projects as needed.

The Figures 3.6 and 3.7 below describe the interaction between the Experience Factory and the project organization.

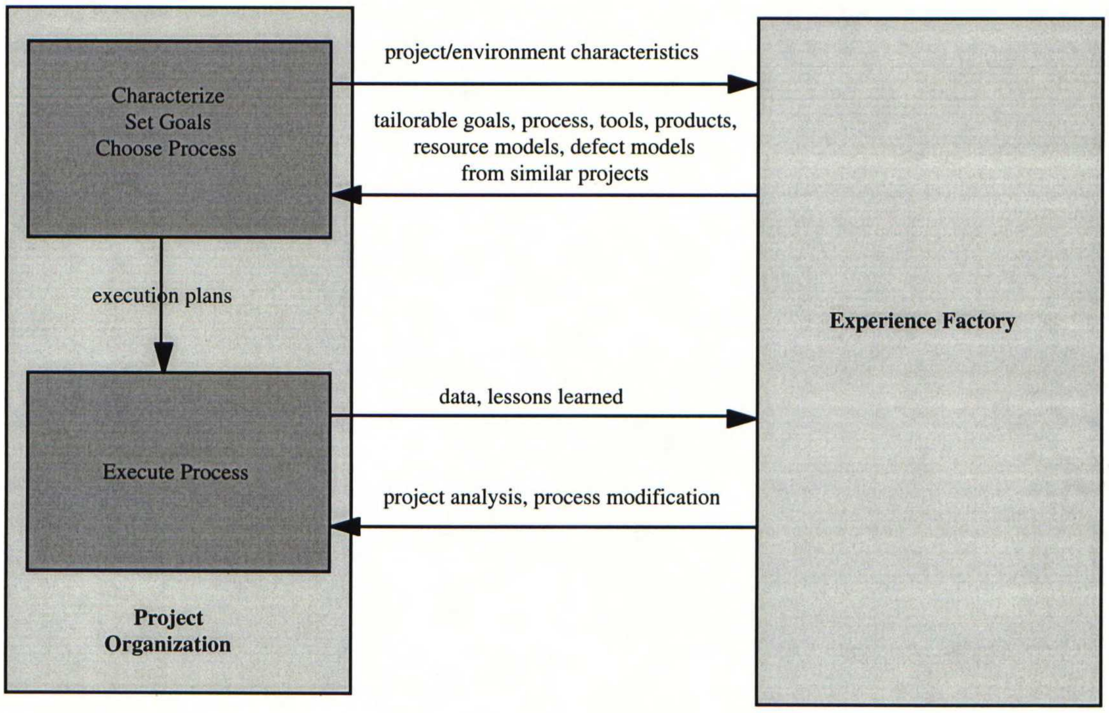

Characterize
Set Goals
Choose Process
project/environment characteristics
tailorable goals, process, tools, products, resource models, defect models from similar projects
execution plans
Experience Factory
data, lessons learned
Execute Process
project analysis, process modification
Project Organization

Figure 3.6. Role of the Project Organization [BaMc95].

Projects choose and tailor project specific process, based on project goals and characteristics, using process assets from the Experience Factory. Results of the tailoring activity are documented to execution plans, for example, project plan. During the project life time results of process execution are reported to Experience Factory, for example, as metrics. Project specific process can be modified during the project life cycle, if it seems necessary based on the analyzed project data.

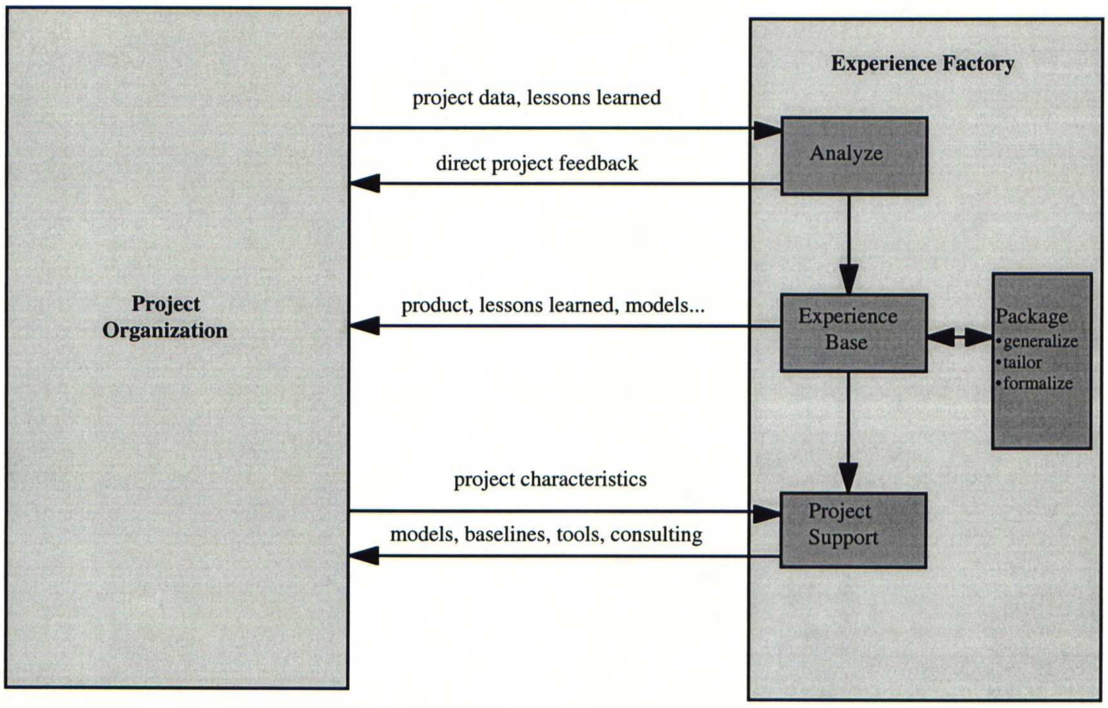


Figure 3.7. Role of the Experience Factory [BaMc95].

Experience Factory gets data and lessons learned from project. Experience Factory organization analyzes the data and provides direct feedback to the project. Feedback can include also process modification suggestions.

Project experiences are packaged to experience base trough generalization, tailoring and formalization. The experience base contains an accessible and integrated set of analyzed, synthesized, and packaged experience models that capture past experiences [BaCR94b].

Experience Factory organization provides support for projects in the definition of project specific process using the packaged project experiences.

3.2 Problems in Organizational Software Process Improvement and How PSP Can Help?

One of the key challenges in software process improvement is to make it reach the individuals, especially to software engineers doing their daily software development work. Often software process improvement (SPI) programs and methods to implement them, are represented by the organization point of view, having strong management emphasis, and considering the implementation as one massive effort in the whole organization. Software engineers see lots of confusion and adding of bureaucracy into their lives without any visible benefit to them personally. As a consequence they often half-heartedly contribute to the program and with this kind of motivation the quality of collected data is not very high, let alone it would improve over time.

PSP is aimed at software engineers. It deals with the topics in organizational process improvement, but does it purely individual point of view. It brings in and implements on individual level those management practices which the organization management can built onto. In addition it does it in such a way that there is immediate benefit to individuals themselves, this giving motivation to do it well. The data is reliable. And when the personal experience grow the reliability of data will also grow.

PSP is build into a shape of the training program. It serves well the training of newcomers into process issues, doing it in terms that make sense to them. In many software organizations with more experienced people, PSP could be the means to bring in process improvement terms and practices for them, too. It might take some adjustment to basic PSP training process, like making it more adjusted to the daily work and technology environment of the organization, but the concept is still valid

Obviously, PSP practices do not integrate automatically into the organization management process. On the contrary, it takes significant effort to align all elements in individual and different organization levels to serve the consistent daily management of software development. Partly for this purpose, SEI (Software Engineering Institute) is developing also TSP (Team Software Process). Serving higher management levels will also require different tools to collect and combine results for organization use, compared to that what might serve individuals. Also, the organization level must produce feedback to individuals, in addition to that what individuals collect for themselves. Management must show the results and show the decisions that were made based on those results, that is, show the value of individual work in the whole organization.

On the other hand, individual level is a vital part of getting the whole management structure work. The software business management needs the basic data from software engineers. So needs the process improvement too, in order to be able to show clearly how successful it has been.

In short one could say that PSP is excellent training program to lead new and even more experienced software engineers into the world of continuous software process improvement, but even more importantly it is a means to implement the individual level of software management structure. But like every other model, when used in any organization, also PSP must always adjusted to the specific needs of that organization.

4. Reported experiences and results of teaching and using the Personal Software Process (PSP)

4.1 An Empirical Study of the Impact of PSP on Individual Engineer

The SEI technical report, *The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers* [HaOv97], presents the results of a study on how the PSP affects the work of software engineers [HavOv97]. The data analyzed in this report are derived from 23 offerings of the PSP training course. The course is aimed at software engineering practitioners. The report describes the effect of PSP on key performance dimension of 298 software engineers. In this study five personal process improvements dimensions of the PSP, i.e., size and effort estimation accuracy, product quality, process quality, and personal productivity, were examined. The study results are briefly summarized below: [HavOv97]

- Effort estimates improved by a factor of 1.75 (median improvement). See Table 4.1.
- Size estimates improved by a factor of 2.5 (median improvement). See Table 4.2.
- The tendency to underestimate size and effort was reduced. The number of overestimates and underestimates were more evenly balanced.
- Defects found in the product at unit test, improved 2.5 times (median improvement). The quality improvements are illustrated in the table 4.3 below with data from a single engineer.
- Process quality, the percentage of defects found before compile, increased by 50 % (median improvement).
- Personal productivity, lines of code produced per hour, did not change significantly. However, the improvement in product quality resulting from the PSP is expected to improve productivity and cycle time as measured at the project level (i.e., when integration and system test phase effort are included in productivity and cycle time).

Assigment	Estimated Effort	Actual Effort	Effort Estimation Accuracy
1	240	314	-31%
2	180	205	-14%
3	160	312	-95%
4	252	282	-12%
5	300	378	-26%
6	600	419	30%
7	290	313	-8%
8	180	192	-7%
9	510	535	-5%

PSP0 aggregate effort estimation accuracy = -43.38 %
PSP2 aggregate effort estimation accuracy = -6.12 %
Effort estimation accuracy improvement factor = 7.1

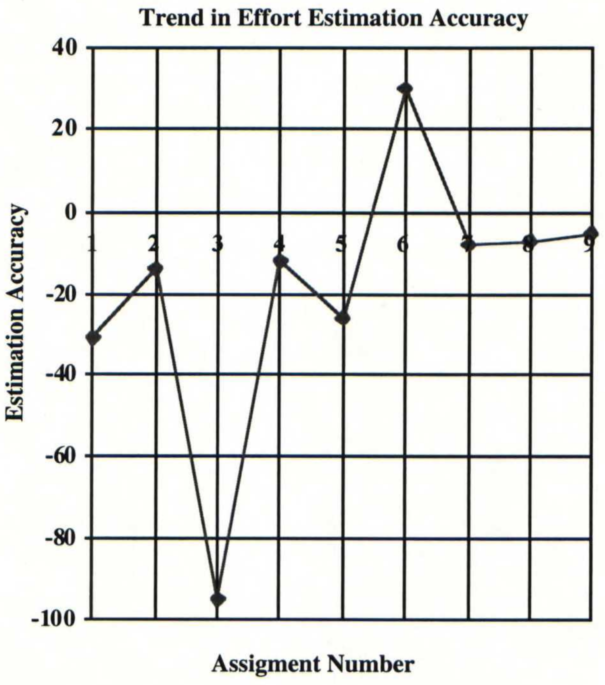


Table 4.1. Sample Data for Effort Estimation [HaOv97].

Assigment	Estimated Effort	Actual Effort	Effort Estimation Accuracy
1	240	314	-31%
2	180	205	-14%
3	160	312	-95%
4	252	282	-12%
5	300	378	-26%
6	600	419	30%
7	290	313	-8%
8	180	192	-7%
9	510	535	-5%

PSP0 aggregate effort estimation accuracy = -43.38 %
PSP2 aggregate effort estimation accuracy = -6.12 %
Effort estimation accuracy improvement factor = 7.1

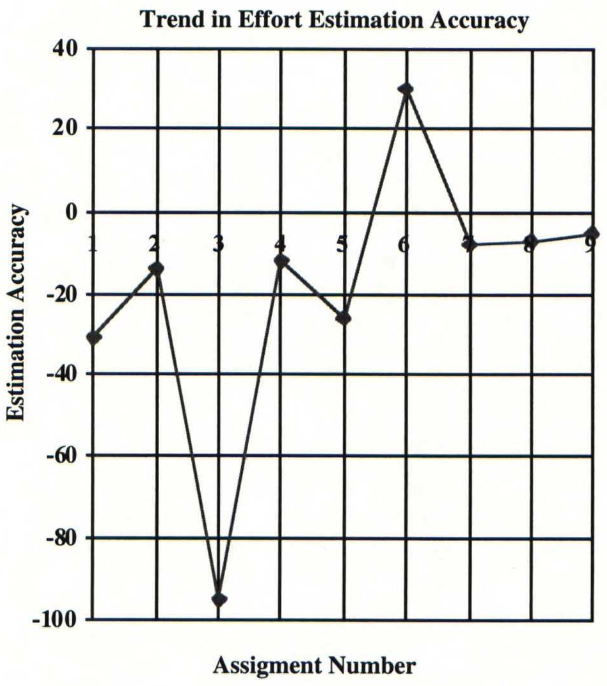


Table 4.2. Sample Data for Size Estimation [HaOv97].

Assigment	Total Defects per KLOC	Defects per KLOC Removed in Compile	Defects per KLOC Removed in Test
1	108.43	60.24	48.19
2	101.27	37.97	63.29
3	85.11	47.87	37.23
4	58.82	39.22	19.61
5	100.00	63.64	36.36
6	45.45	39.77	5.68
7	53.57	8.93	0.00
8	27.03	0.00	9.01
9	15.34	6.13	3.07

	PSP0 Defect Density	PSP2 Defect Density	Improvement Factor
Total Defect Density	94.29	25.50	3.7
Compile Phase	48.57	5.46	8.9
Test Phase	45.71	3.64	12.6

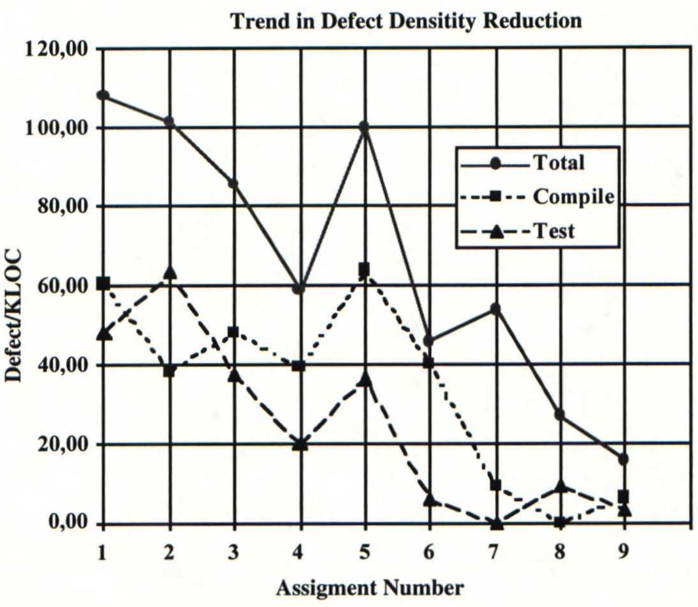


Table 4.3. Sample Data for Defect Density [HaOv97].

4.2 Academic Experiences Using the PSP

Most of the data on the PSP methodology's successes are based on academic classroom experiences at Carnegie Mellon University, but a growing number of universities have started to teach the PSP, such as Embry-Riddle Aeronautical University [ToHi97], University of Utah [Will97], Lawrence Livermore National Laboratory (LLNL), California [Roy96], and University of Galgary [Smith97].

Embry-Riddle Aeronautical University started to introduce software process concepts and activities into their two first year computer science courses (CS1 and CS2) during academic year 1995-96. They used an early draft version of Watts Humphrey's *Introduction to the Personal Software Process* [Hump97b] as a training material. They started the project called "Doing Quality Work" (DQW/PSP). The project goal was to emphasize to the students, from very beginning, how important quality was in their work.

The following list summarizes the conclusions of the DQW/PSP project: [ToHi97]

- Humphrey's introductory text *Introduction to the Personal Software Process* [Hump97b] provides an excellent foundation and motivation for teaching DQW/PSP concepts.
- There is a need for simplification and automation of forms, logs and records keeping.
- DQW/PSP concept need to be integrated throughout the computer science curriculum.
- DQW/PSP data provides valuable information to teachers for analyzing student effectiveness in completing course work, especially programming projects.

During the fall semester 1995 (from September to December) Margaret A. Ramsey taught the PSP at New Jersey University. The course was taught using *A Discipline for Software Engineering* [Hump95a] as the text book. The course was open to Computer Science seniors and graduate students. The class began with enrollment of fifteen students, the major of whom were seniors who had not worked extensively in the computer field.

The following list summarizes the conclusions of their academic course experiences: [Rams96]

- The software engineering work experience of the studendts was limited, so it was very difficult to get them to participate in any classroom discussion and interaction. It also made class motivation more difficult.
- Students were concerned about how the PSP grading would be done and how to know the "right" answer because of non-traditional course structure. The course was very different from any other computer course they had taken earlier.
- Students either did very well (eight students) or did not finish during semester. Four students took incomplete grades and three dropped the course. Those that

attended regularly and completed the work expressed a feeling of accomplishment and the desire to continue to use the tools and ideas they had learned.

- The PSP course focus and level are well suited to the Computer Science senior and graduate student level.
- The PSP course instructor and teaching assistant may need to have side lectures focusing on the statistical behind the course. The lecture time was not for adequate remedial statistics.

Based on University of Galgary PSP course experiences during winter term 1996, they learned that data gathering for the PSP was tedious. It is possible that time spent in phases as the postmortem could be saved through automating all or some of the parts of the PSP data gathering. It may not be viable to automate the entire PSP, but most important things to automate are the data entry, calculation, and analyses. At minimum, the Project Plan Summary, time recording, defect recording, and the PIP should be automated. [Smith97]

Humphrey claims [Hump98d], that the best evidence of how well the PSP course works is in how it changes the way students spend their time. As shown in Figure 4.1, at the beginning of the PSP course students spend less time on design than on any other activity. At the end of the course, they typically spend more time on design than on coding, compiling, or testing. This is what Humphrey, and the SEI have been trying to get software engineers to do for years.

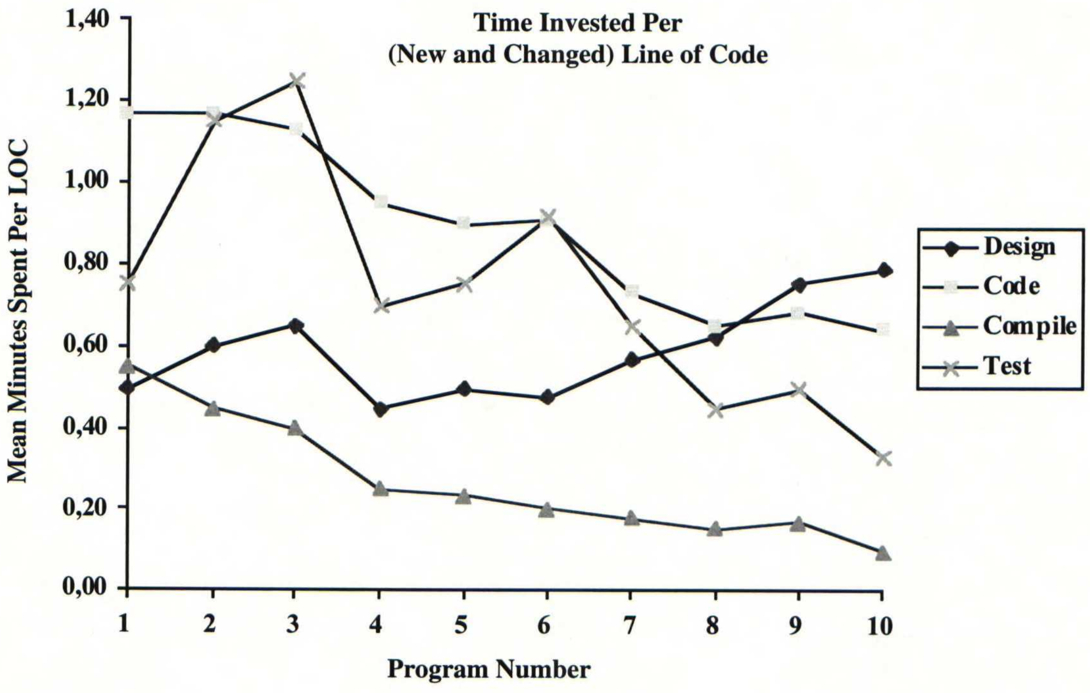


Figure 4.1. Effort distribution results [Hump98d].

4.3 Industrial Result with the PSP

While there is extensive data on PSP in the classroom, there is less on how PSP helps the industry. A number of industrial groups are experimenting with the PSP methods, however statistics on its benefits are limited. The article *¨Result of Applying the Personal Software Process¨* [FeHu97] reports the experience of three industrial software groups that have used PSP and collected data to show its effectiveness. These groups are Advanced Information Services (AIS) Inc., Motorola Paging Products Group, and Union Switch & Signal Inc.. Each has trained several groups of engineering and measured the results of several projects that used PSP methods. In all cases, the projects were part of the companies´ normal operations.

Advanced Information Services (AIS) is located in Peoria, Illinois. The company offers software development, process training, Internet services, and consulting services. The company has trained its engineers in the PSP. AIS first introduced PSP with a pilot course in the spring of 1994. Figure 4.2 shows the estimated and actual times AIS engineers took to development nine components of a moderate-sized product. It shows the estimating error for each of these components. Each component had about 1000 lines of code. The three bars on the left of the chart show the engineer's time estimates for the weeks it would take to develop the first three components. The average estimating error for these first three components was 394 %. After PSP training these same engineers completed the remaining six components. The average estimating error for the last six components was -10.6 % [Hump98b]. For example the original estimate for component 8 was 14 weeks and the work was completed in 14 weeks. Figure 4.3 shows the estimated and actual weeks to develop nine components of a product.

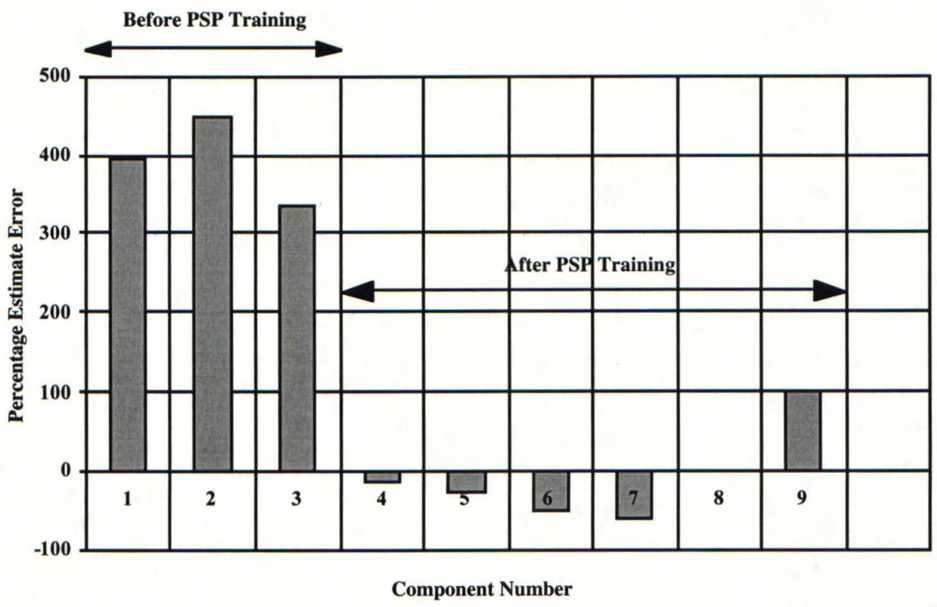


Figure 4.2. AIS Error in Estimated Weeks of Work [FeHu97].

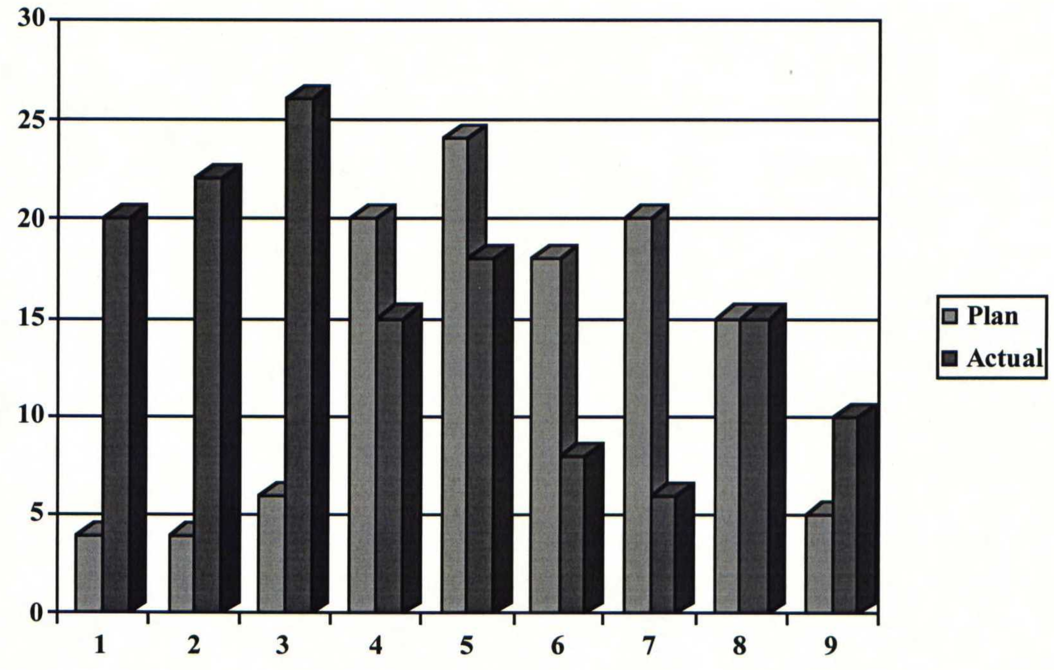


Figure 4.3. AIS Estimated Weeks of Work [FeHu97].

Table 4.1 shows data on six projects at Advanced Information Service (AIS) (an early PSP user). Here, program size is measured in lines of code (LOC) or the requirements count (Req.). Three of these projects used PSP-trained engineers and three used either engineers untrained in the PSP or a mix of trained and untrained engineers. Of the projects that used only PSP trained engineers, all were delivered on or ahead of schedule, and only one defect was found in acceptance testing. None exhibited any defects during several months of customer use.

Project	Staffing PSP	Staffing Non-PSP	Product Size	Delivery Months Plan/Actual	Acceptance Test Defects	Usage Defects
A	3	-	24 Req.	7/5	1	0
B	-	3	19 Req.	2/5	11	1
C	-	3	30 Req.	10/19	6	14
D	1	-	2233 LOC	6/6	0	0
E	1	-	1400 LOC	2/2	0	0
F	2	1	6196 LOC	2/2	0	3

Table 4.1. Summary of AIS Project Data [FeHu97].

"All engineers and managers at the AIS subsidiary in India have completed PSP training 1997. In the USA, 58 percent of engineers and managers have completed training during 1997. AIS now trains all new engineers in PSP before they are assigned a project." [FeHu97]

The Figures 4.4 and 4.5 below shows, what the engineers from Motorola Paging Group think about the PSP, and how their work habits are changed after PSP education five months later.

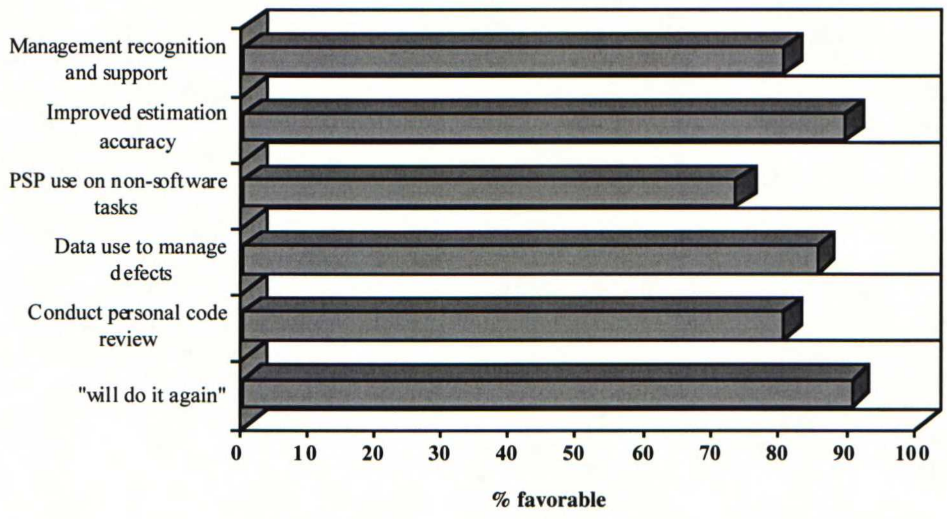


Figure 4.4. What do the Engineers from Motorola Paging Products Group think about PSP after education [MaKN96]?

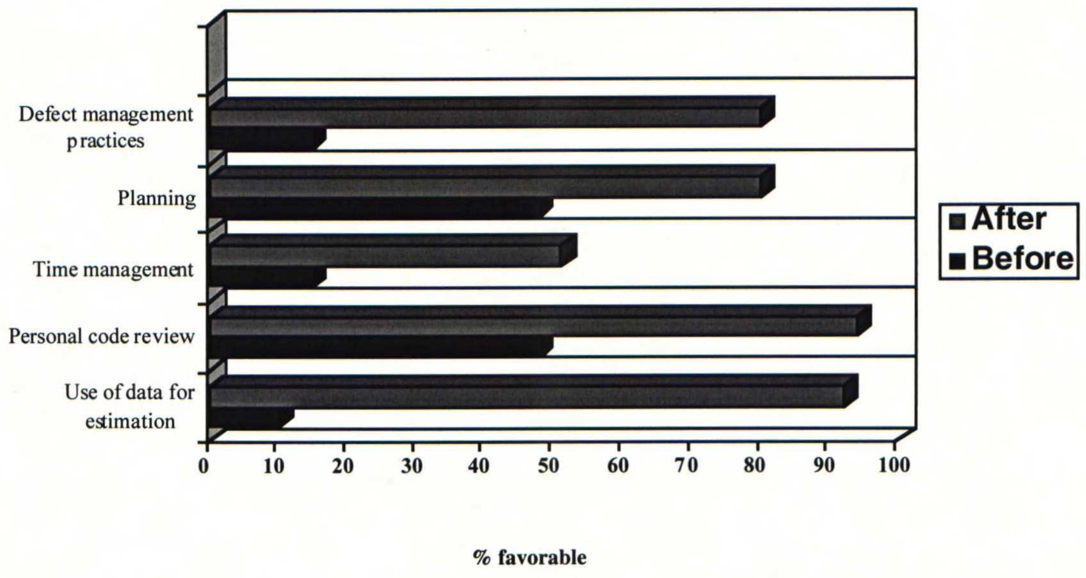


Figure 4.5. The Engineers Work Habits from Motorola Paging Products Group

A study of the implementation of some PSP concepts in a commercial organization is reported in the paper, *Implementing Concepts from the Personal Software Process in an Industrial Setting* [EmSM96]. It describes the general lessons learned during the implementation of PSP.

Some of the lessons from this paper are summarized below: [EmSM96]

- Lectures should cover all the typical personal life cycles that are in effect in the organization, not only the one presented in the PSP manuscript. This makes the classroom teaching more relevant to the participants′ real programming tasks.
- It is important to customize the PSP data collection forms to the personal processes of individual participants. They found, for example, that personal processes varied depending on which product line the programmers were working on.
- All forms must be piloted with the participants in their real work environment. Even if forms were designed to fit their personal processes, actual use in realistic programming tasks may reveal deficiencies in the design of the forms.

- It would be better to give the supervisors of the participants at least a formal short overview of PSP so they understand it and see its benefits. This would help gain stronger commitment for PSP.
- It is important to have automated tools that support the participants′ data collection and also data analysis. Furthermore, it would be preferable if the data collection forms are available in editable format for the participants so that they can customize the forms themselves as they gain a better understanding of their processes.

5. THE PILOT PSP COURSE AT HELSINKI UNIVERSITY OF TECHNOLOGY (HUT)

5.1 The Pilot PSP Course Overview

The goal of the PSP is to make students aware of the process they use to do their work and of their performance of those processes. Individual students are trained to set personal goals, define the methods to be used, measure their work, analyze the result, and adjust their methods to meet their goals. The concepts, structure, and activities of the PSP, what we used at HUT, are described in detail in the textbook, *"Introduction to the Personal Software Process"* [Hump97b]. The material was presented as lectures, discussions and workshop exercises. The textbook contents are designed to be covered in two semesters. In the first part, *Personal Time Management*, students learn to collect fundamental data, estimate software size and effort, follow plans and track progress. In the second part, *Personal Quality Management*, students learn to practice quality management techniques, use design and code reviews and use yield management to improve quality.

Most software is developed by groups. One or more teams are given responsibility for carrying out the activities of the software process. The Personal Software Process by Watts Humphrey focuses on individual software process. One foundation for productive participation in a software development team is a refined personal process.

At Helsinki University of Technology we decided to test the PSP methods and forms on the graduate Tik-76.115 Software Project course. "During this two-term, five credit course, students work through major software projects in groups of five to seven persons. Each project comprises all typical software design and implementation phases, such as requirements analysis, conceptual and detailed design, coding, testing, documentation, and delivery to the customer. The course emphasizes well-planned and implemented software engineering methods, project management, team management and system documentation [AlVa98]."

The PSP course introduced students to disciplined methods for managing and planning their work, and to demonstrate, with their own data, the value of these practices. All students had to do their own work, but they did not have to work alone. Students had to make their own estimates, do their own designs, code their own programs, and compile and test their own work.

Quality methods take time to learn and practice, so the principal focus in our PSP course was to test how the students can use some PSP methods on their own software project and what were their experiences of the PSP.

5.2 The Pilot PSP Course Elements

This chapter provides an overview of the basic PSP forms and elements that were required on the pilot PSP course. This information is provided to give the reader some context for data that was analyzed for this thesis. The following basic PSP elements were required:

- Time Recording Log
- Defect Recording Log
- Weekly Activity Summary
- Job Number Log
- Project Plan Summary
- Personal Checklist

5.2.1 The Time Recording Log

There are three basic measures in the PSP: development time, program size and defects. All other PSP measures are derived from these three basic measures.

The goal of the PSP time measures are to determine how much time students spend in each PSP phase and help them to make better time estimates. Minutes are the unit for development time measure. A form, the Time Recording Log, is used to record development time. Students track the number of minutes they spend in each PSP phase. They also follow how much time goes to interruptions such as phone calls, people wanting to chat, coffee breaks etc.. The interruptions are recorded in the Time Recording Log column Interruption time.

An example of how the Time Recording Log is completed is shown in Table 5.1. In the example, the student Y started the Class activity of his project on November 11 at 16:00 and finished the class at 16:40. His first activity was to attend a PSP lecture. In this time the elapsed time and Delta Time were same, 40 minutes, because he had not any interruptions. The remaining activities, Project Management, Meeting, Planning etc., are recorded in similarly.

Student: Y	Date: 7/12/97
Instructor: Ms. Virpi Kaltio	Class: The Pilot PSP Group

Date	Start	Stop	Interruption Time (min)	Delta Time	Activity	Comments
5.11.97	16:00	16:40		00:40	CLASS	PSP lecture
	20:50	21:35		00:45	PROJECT MANAGEMENT .	Review preparation
6.11.97	09:20	10:05		00:45	MEETING	Review
12.11.97	21:13	21:48		00:35	PLANNING	Technical spesification
13.11.97	15:10	16:30		01:20	MEETING	Team meeting
17.11.97	21:35	23:04		01:29	PLANNING	Technical spesification
18.11.97	18:15	20:00		01:45	PLANNING	Technical spesification
	22:26	00:08		01:42	DOCUMENT	Technical spesification
19.11.97	15:00	17:20		02:20	PLANNING	Technical spesification
24.11.97	12:15	12:45		00:30	CLASS	Proto lecture
	13:00	13:30		00:30	MEETING	Team meeting
	20:26	21:41		01:15	DOCUMENT	Technical spesification
	21:51	22:15		00:24	ADP	
25.11.97	15:00	16:40		01:40	MEETING	Customer meeting
	21:01	22:41		01:40	DOCUMENT	Technical spesification
	22:41	23:12		00:31	ADP	Documents' outlook improvement
26.11.97	22:32	23:28		00:56	DOCUMENT	Technical spesification
27.11.97	15:05	16:10		01:05	MEETING	Team meeting
	16:10	17:05		00:55	PLANNING	Technical spesification
30.11.97	16:11	18:45	30	02:04	DOCUMENT	Technical spesification
	21:06	22:15		01:09	DOCUMENT	Technical spesification
1.12.97	13:17	15:40		02:23	PLANNING	Technical spesification
	19:25	20:17		00:52	MEETING	Team meeting
	22:02	23:55		01:53	DOCUMENT	Technical spesification
4.12.97	11:17	11:54	5	00:32	ADP	
	11:55	12:02		00:07	DOCUMENT	Test plan
	12:29	12:50		00:21	DOCUMENT	Test plan
5.12.97	15:05	16:00		00:55	MEETING	Test plan
7.12.97	16:42	20:42		04:00	DOCUMENT	Technical spesification
	23:10	23:31		00:21	CLASS	Filling of PSP forms
	23:31	00:07		00:36	PROJECT MANAGEMENT .	Status report
Cumulative Total:				36:00:00		

Table 5.1 Student Y´s Time Recording Log Example.

5.2.2 The Weekly Activity Summary

The Weekly Activity Summary form was used for tracking and analyzing the time spent on the PSP course. This data is summarized from the Time Recording Log. In the beginning the students track their time using the Time Recording Log. After gathering a week or two of time data, students summarized the data in a more useful form. An example of a partially completed Weekly Activity Summary is shown in Table 5.2.

This example shows a record of the time student Y spent on each principal activity during each day of the previous week. Previous Week´s Time shows the average, maximum, and minimum times he spent on each task category during the earlier weeks of the semester. Current Weeks Times shows the total, average, maximum, and minimum times he spent in each work category for the entire semester so far, including the latest week.

The purpose of the Weekly Time Summary is to help students to understand how they spend time, and plan the time they will spend in the future.

Name: Student Y Date: 16/2/98

Instructor: Ms. Virpi Kaltio Class: The Pilot PSP Group

Task		Class	Study	Meeting	Project Manag.	Plan	Code	Test	Doc.	ADP	Total
Date											
M	9.2					107	151				258
T	10.2		90								90
W	11.2										0
T	12.2		18			38	136				192
F	13.2					63	69				132
S	14.2				19		195				214
S	15.2		16						241		257
Total			124		19	208	551		241		1143

Period Times and Rates

Number of Weeks (prior number +1) 19

Previous Week's Time

Total	229	478	1422	290	1263	1813		1829	752	8076
Avg.	13	27	79	16	70	101		102	42	449
Max.	129	151	263	68	319	1349		780	262	1664
Min.	30	20	45	22	35	29		84	32	115

Current Week's Time

Total	229	602	1422	309	1471	2364		2070	752	9219
Avg.	12	32	74	16	77	124		109	40	485
Max.	129	151	263	68	319	1349		780	262	1664
Min.	30	20	45	19	35	29		84	32	115

Table 5.2. Student Y's Weekly Activity Summary.

5.2.3 The Job Number Log

The Job Number log form is used to track the job numbers for each project, and also records key information on each project. This form is designed to record estimated and actual time data. This log is a product planning document since it deals with product data. "*A product plan* is a plan for producing a product. This could be a program, a report, or even something simple like reading a textbook chapter. The key point is that some tangible result is produced [Hump97d]." The Time Recording Log and Weekly Activity Summary contain data on weekly periods. They are thus period planning documents. "*A period plan* is a plan for how you will spend your time for some period of time. Example periods would be a day, week, or semester [Hump97d]."

An example of student Z's Job Number Log is shown in Table 5.3. He tracked only one project activity: writing text. In the example, student Z wrote the project plan of his project on October 8. His time estimate for this task was 360 minutes. At the end of writing, he entered the actual time 245 minutes, and the actual units seven pages. The Actual Rate is the Actual Time divided by Actual Units. For job one the Actual Rate is 245/7 = 35 minutes per page. At the end of the job the student calculated and entered the To Date Time for all the tasks done to date of the same process type. In this case 655 minutes Actual Time for job 5, the To Date Time for jobs 1 through 4 as 1355, and the To Date Time for writing text 5 is 1355 + 655 = 2010 minutes. Student Z entered the To Date Units for all the tasks completed. In this case 16 pages Actual Units for job 5, To Date Units for jobs 1 through 4 as 28, and the To Date Units for writing text 5 is 28 + 16 = 44 pages. The To Date Rate is the To Date Time divided by To Date Units, i.e., 2010/44 = 46 minutes/page.

The purpose of the Job Number Log is to help students to understand how much time each of their jobs has taken, and plan how much time they will spend on similar tasks in the future.

Name: Student Z Date: 25/05/98

Instructor: Ms. Virpi Kaltio Class: The Pilot PSP Group

Job #	Date	Pro-cess	Estimated		Actual			To Date				
			Time	Units	Time	Units	Rate	Time	Units	Rate	Max	Min
1	Description:		Project Planning			(unit = pages)						
	08.10	text	360	1	245	7	35	245	7	35	35	35
2	Description:		Test Planning			(unit = pages)						
	08.12	text	600	1	435	10	44	680	17	40	44	35
3	Description:		Manuscript			(unit = pages)						
	14.02.	text	1200	1	485	8	61	1165	25	47	61	35
4	Description:		Raporting			(unit = pages)						
	25.03.	text	600	1	190	3	63	1355	28	48	63	35
5	Description:		Demo			(unit = pages)						
	23.04.	text	600	1	655	16	41	2010	44	46	63	35

Table 5.3 Student Z´s Job Number Log.

5.2.4 The Project Plan Summary

Project summary data is recorded on the Project Plan Summary form. The Project Plan Summary form contains spaces for planning the job in advance, recording the size of the finished job, and recording the time spent by project phase. Table 5.4 shows the five sections of project plan summary that were used: Summary, Program Size, Time in Phase, Defects Injected, and Defects Removed. This form is very complicated and difficult to explain shortly. More complete information on the contents of the PSP Project Plan Summary form is provided in The Appendix C: The PSP Project Plan Summary Instructions.

Student:	Student Q		**Date:**	15.3.1998	
Program name:			**Program #:**		**2**
Instructor:	Ms. Virpi Kaltio		**Language:**		**C**
Summary	**Plan**	**Actual**	**To Date**		
Minutes/LOC	**9,33**	**0,00**	**3,86**		
LOC/Hour	**6,43**	**0,00**	**15,56**		
Defects/KLOC	**333,33**	**181,82**	**214,29**		
Yield	**0,00**	**0,00**	**0,00**		
A/FR.	**1,00**	**5,00**	**3,00**		
Program Size (LOC)	**Plan**	**Actual**	**To Date**		
Total New and Changed	6	11	**14**		
Maximum Size	165				
Minimum Size	180				
Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date%**	
Planning	**8,00**	2	**6**	**11,11**	
Design	**4,00**	5	**7**	**12,96**	
Code	**40,00**	13	**33**	**61,11**	
Code Review	**2,00**	5	**6**	**11,11**	
Compile	**2,00**	1	**2**	**3,70**	
Test	**0,00**		**0**	**0,00**	
Postmortem	**0,00**		**0**	**0,00**	
Total	**56**		**54**	**100,00**	
Maximum Time	**1540**				
Minimum Time	**1680**				
Defects Injected	**Plan**	**Actual**	**To Date**	**To Date%**	**To Date Def./ Hour**
Planning	**0**		**0**	**0,00**	
Design	**0**	1	**1**	**33,33**	**8,57**
Code	**2**	1	**2**	**66,67**	**3,64**
Code Review	**0**		**0**	**0,00**	
Compile	**0**		**0**	**0,00**	
Test	**0**		**0**	**0,00**	
Total	**2**	**2**	**3**	**100,00**	
Defects Removed	**Plan**	**Actual**	**To Date**	**To Date%**	**To Date Def./Hour**
Planning	**0**		**0**	**0,00**	
Design	**0**		**0**	**0,00**	
Code	**0**		**0**	**0,00**	
Code Review	**0**		**0**	**0,00**	**0,00**
Compile	**2**	2	**3**	**100,00**	**90,00**
Test	**0**		**0**	**0,00**	**0,00**
Total	**2**	**2**	**3**	**100,00**	

Table 5.4 The Student´s PSP Project Plan Summary Example.

5.2.5 The Defect Recording Log

The goals of the PSP defect measures are to provide a historical baseline of defect data and understand the numbers and types of defects injected. "The term *defect* refers to something that is wrong with a program, such as a syntax error, a misspelling, a punctuation mistake, or an incorrect program statement. Defects can occur in programs, in designs, or even in the requirements, specifications, or other documentation. A defect is thus an objective thing. It is something you can identify, describe, and count [Hump97b]." Defects are recorded on the Defect Recording Log as they are found and fixed. The example Defect Recording Log illustrates how this form is used in Table 5.2. It shows the information that is recorded for each defect: the date, sequence number, defect type, phase in which the defect was injected, phase in which it was removed, fix time, and the a description of the problem and fix. The *fix time* is the time, in minutes, spent finding and fixing the defects.

In analyzing defects, it is helpful to divide them into categories. Each defect is classified according to a defect type standard. *"Introduction to the Personal Software Process"* [Hump97b] classifies defects into ten general types. This standard was modeled on the work of Chillarege at IBM Research, and it should be sufficiently general to cover most needs [Hump95a]. The defect types from the Defect Type Standard are presented in Table 5.5.

In the example Defect Recording Log (Table 5.2), student X found the first defect on January 17. The defect was of type 50 (interface) that was injected during the code phase and removed during compile. Student X spent one minute finding and fixing the defect. The second error was of type 10 (documentation) that was injected during the code phase and removed during review, and took 17 minutes to find and fix it.

Name: Student X Date: 7/12/97

Instructor: Ms. Virpi Kaltio Class: The Pilot PSP Group

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
17.1.98	1	50	code	compile	1	

Descr. Wrong number of prameters for the function FillUsers.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
21.1.98	2	10	code	review	17	

Descr. Incorrect format of JavaDoc comments.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
21.1.98	3	20	code	review	1	

Descr. Missing half-point

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
23.1.98	4	20	code	compile	1	

Descr. Incorrect function name (typo).

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
23.1.98	5	20	code	compile	1	

Descr. Missing half-point

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
23.1.98	6	40	code	compile	2	

Descr. Same variable declarated two times in a function.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
25.1.98	7	20	code	compile	1	

Descr. Missing half-point

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
11.3.98	11	20	code	review	1	

Descr. The bracket is missing at the end of function.

Table 5.2. Student X's Defect Recording Log Example.

Defect Type Number	**Defect Type Name**	**Description**
10	Documentation	comments, messages
20	Syntax	spelling, punctuation, typos, instruction formats
30	Build, package	change management, library, version control
40	Assignment	declaration, duplicate names, scope, limits
50	Interface	procedure calls and references, I/O, user formats
60	Checking	error message, inadequate checks
70	Data	structure, content
80	Function	logic, pointers, loops, recursion, computation, function defects
90	System	configuration, timing, memory
100	Environment	design, compile, test, other support system problems

Table 5.3. Defect Type Standard [Hump95a].

6. Experiences of the Pilot PSP Course at Helsinki University of Technology (HUT)

6.1 The Survey of the Pilot PSP Course Experiences

This chapter presents the results of the pilot PSP course feedback questionnaire for 9 students who took the PSP pilot course at Helsinki University of Technology during fall 1997 and spring 1998. The purpose of the questionnaire was to gather relevant information from the students′ subjective PSP course experiences and to collect ideas for future improvement of the PSP course. The *Feedback Questionnaire - Tik-76.161 Individual Project (3 cr.) P* form is shown in Appendix 5. Several questions in this questionnaire ask about usefulness using a four level scale. In each question the students considered following two aspects:

- How well it teaches the purpose and practices of the subject matter?
- How beneficial is it for the actual SW development?

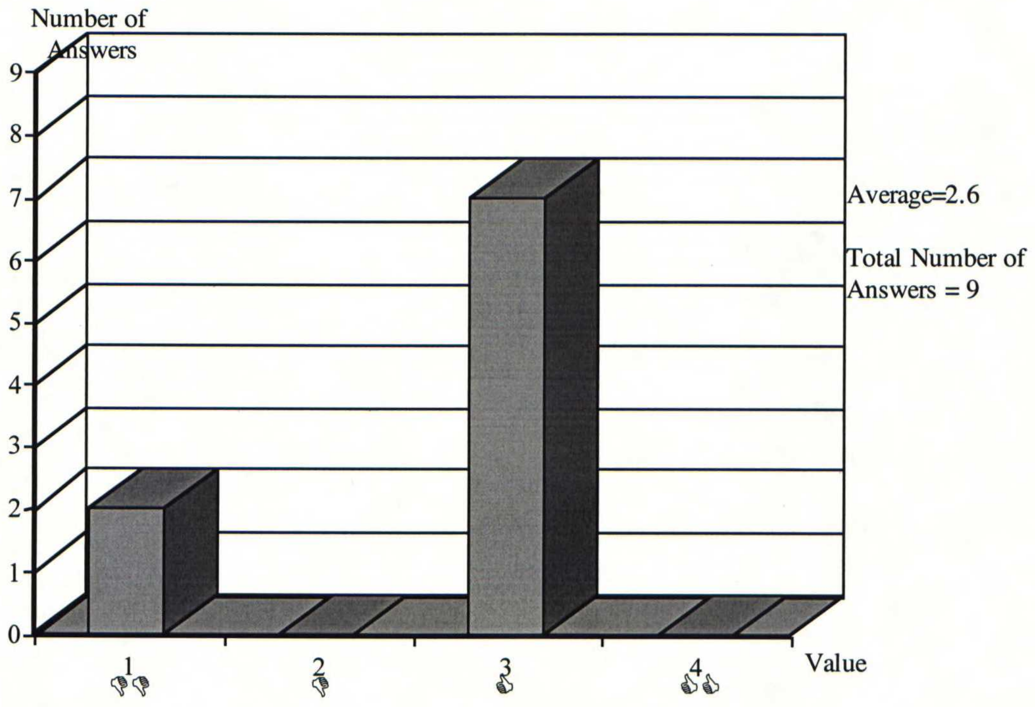
In the last section of the questionnaire students were asked to answer more general questions about the pilot PSP course.

The results of the survey are presented in the following chapters 6.2 - 6.4.

6.2 Questions Related to Time Management

6.2.1 The Time Recording Log

Question 1: How useful the Time Recording Log was in planning and tracking your work in the team?



Question 2: What were the main benefits in the use of Time Recording Log for you and the project team?

Actual answers:

1. I found the Time Recording Log not that useful. The benefits were merely in tracking the time for the reports, not in planning the use of time in advance. I also should have more discipline in using the system. Still logging showed clearly how each individual used time and for what. I cannot state if this system was better than some other. The main benefit was that everybody used the same system.

Some reason why Time Recording Log was not used in its full potential: (1) Our project did not go well so for example I spent less time for the project than anticipated (reasons for this were various). (2) When not doing the project full time, there were no needs to plan in advance. Point is: to get the full benefits of the system it should be applied in cases when one is working in the project more or less full time thus using it in time planning (3) One should have more categories of different work tasks. (*Role or roles in the project team: Testing, coding and planning*)

2. Maybe it was a little bit easier to predict the amount of work for the next phase of the project. (*Role or roles in the project team: Documentation)*
3. Better understanding of how much time things take. (*Role or roles in the project team: Project manager)*
4. I got exact data about the time I spend to different parts of work. (*Role or roles in the project team: Project manager and programmer)*
5. It was good to see how much time it took to create documents and coding etc.. When I started to do things I preferred to do more because I have to make notes how much I work so I try to gather all jobs that could be done and do them all at the same time. This helps also to concentrate on the project by trying to minimize the interruptions. (*Role or roles in the project team: Project manager)*
6. Personally I think it was most beneficial, because it gave me some idea of what I was spending most of my time. Therefore planning the use of time for successive weeks was easier. (*Role or roles in the project team: Programmer)*
7. Recording the actual working time quite accurately for the course reporting was the main benefit of the Time Recording Log. (*Role or roles in the project team: Coding, customer relations, design)*
8. It was needed in calculating accumulated time usage for each task. (*Role or roles in the project team: Documentation)*
9. It was good way to see what is the real amount of the time that has been spent in this project. (*Role or roles in the project team: Project manager)*

Question 3: How would you improve the Time Recording Log and it's use during the course?

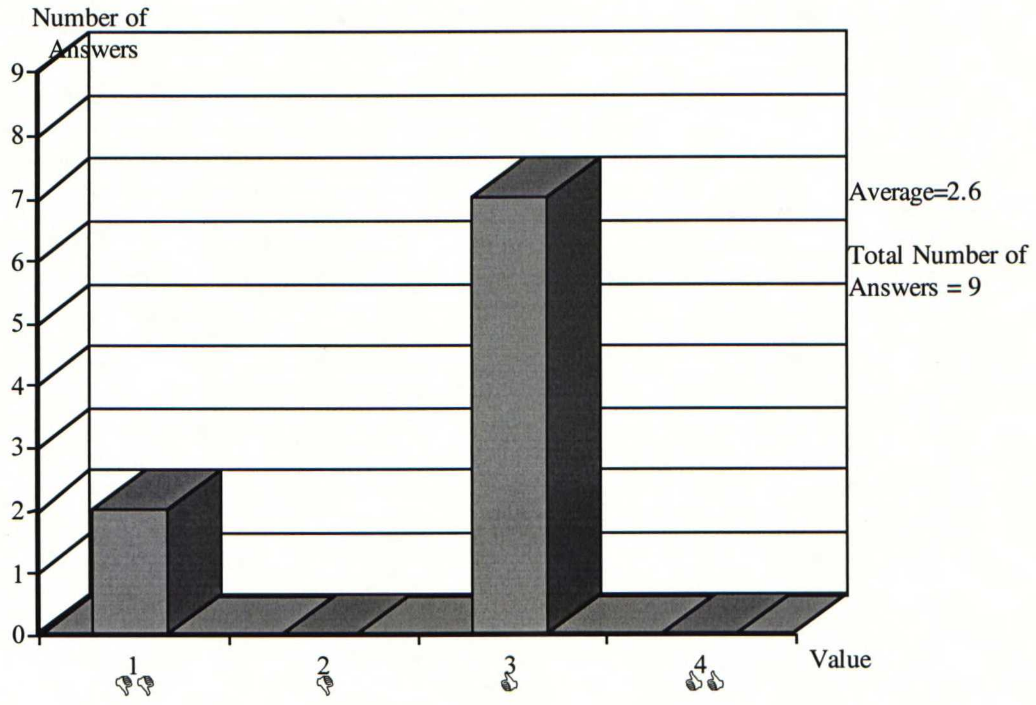
1. Firstly the team should think thoroughly what work categories to use. The rest is up to each individual's discipline in using the system. (*Role or roles in the project team: Testing, coding and planning)*
2. The system must be automatic, or at least semi-automatic. One good solution would be to have a Java applet with which people can actually record transactions during the work day. (*Role or roles in the project team: Documentation)*
3. I used it only for tracking time I put into programming rather than all activities. On the overall time management point of view it would be better to track all time of course, but for the sake of this course concentrating on the SW development work would suffice. (*Role or roles in the project team: Project manager)*
4. We had 8 work types and I found it was too much. Four or even less would have been better. (*Role or roles in the project team: Project manager and programmer)*
5. Using Time Log only in one single separate course is very useless. I think the benefits come when you can see your time usage from yours all studies. Time Logging should be done so that it covers all your actions, related to studies and possibly even related to other matters, too. Then it may become too tedious to do that kind of logging, but it could be done for example so, that you only log time in two phases of the project, prototype 1 and delivery for example. Then you can see real fact how time is spent which helps to improve SW process. (*Role or roles in the project team: Project manager)*
6. The gathering of time data must be made simple, so that the generation of the reports would not take so much time. There was noticeable overhead of writing the reports, which was discouraging at least for me. It was hard to be motivated. The log itself is OK, although the gathering of time data could be automated at least when you are sitting by the computer (some tool, easy to use form or Java based application). (*Role or roles in the project team: Programmer)*
7. During the actual coding period, Time Recording Log was inconvenient because testing, coding and design went hand in hand. So there is room for improvement,

but I dont't know any solutions. Perhaps different categories should be combined. (*Role or roles in the project team: Coding, customer relations and design)*

8. In my opinion, it serves little purpose when used like this. It is more useful when tracking all time usage, i.e., during office hours. The method itself seems sound and useful. (*Role or roles in the project team: Documentation)*
9. I would like to see it little bit more inaccurate (1 minute ->5...15 minutes). Also some regular meetings/checkpoints to check that everybody has still the log faithfully. (*Role or roles in the project team: Project manager)*

6.2.2 The Weekly Activity Summary

Question 1: How useful the Weekly Activity Summary was?



Question 2: What were the main benefits in the use of Weekly Activity Summary for you and the project team?

Actual answers:

1. Weekly Activity Summary showed (sometimes painfully clear) what one had done and what not. It was quite handy in showing the time spent on the project each week. See also comments for the Time Recording Log. (*Role or roles in the project team: Testing, coding and planning)*
2. Afterwards it was like a critic against your usage of time, so the biggest help was for me. It was terrible to follow in what kind of issues your time actually goes. In the end of the project I did not use it for anything else than for course purposes. So, it did not give a good picture about usage of time. (*Role or roles in the project team: Documentation)*
3. I did not use, because I used PSP for about one project per week. (*Role or roles in the project team: Project manager)*
4. It did not give us any more information. It just helped us to count the time we spend to different phases. (*Role or roles in the project team: Project manager and programmer)*
5. I did not find it any good. The work depended from other people, so the work could not be made so regularly, etc. So the weekly report was useless for me. (*Role or roles in the project team: Project manager)*
6. It really helped on planning the use of time for individuals. The use of these was not organized for the whole group. (*Role or roles in the project team:* Programmer*)*
7. I did not find any benefits, personally. (*Role or roles in the project team: Coding, customer relations and design)*
8. It showed the accumulated time for each task. (*Role or roles in the project team: Project manager)*
9. I noted quire clearly that there were weeks without any activities. (*Role or roles in the project team: Project manager)*

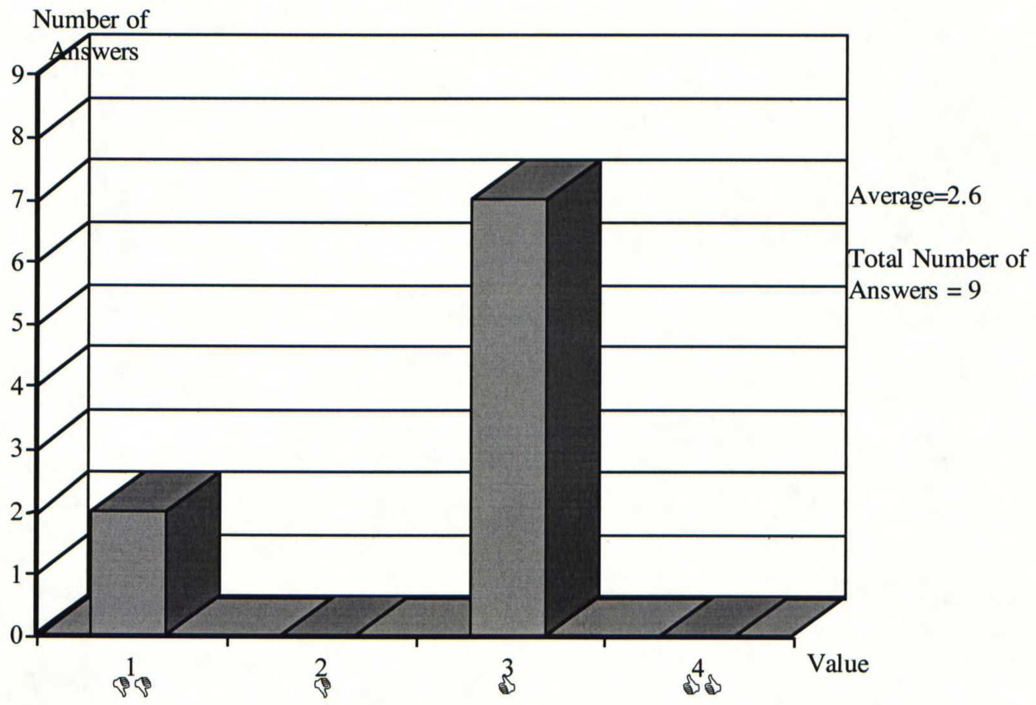
Question 3: How would you improve the Weekly Activity Summary and its use during the course?

Actual answers:

1. The team should think thoroughly what work categories to use. Also for avoiding manual work all reports should be integrated and automated somehow. (*Role or roles in the project team: Testing, coding and planning)*
2. Maybe it must be used for all tasks, not only for this specific project. This way it gives a good picture from time usage overall. (*Role or roles in the project team: Documentation*
3. I didn't use it. (*Role or roles in the project team: Project manager)*
4. Non comments. (*Role or roles in the project team: Project manager and programmer)*
5. Well. Maybe summary could be used as summary for a whole phase. By then it could be watch how time usage depends if the deadline is coming or not and how it makes changes to time usage. (*Role or roles in the project team: Project manager)*
6. Generation of this report should be automated somehow, because all the needed information can be found on the time log. Writing this report from scratch takes some effort and unnecessarily wastes productive time. (*Role or roles in the project team: Programmer)*
7. It could be useful if the tasks under observation were evenly distributed for each week. Now the amount of work used for the course varied greatly in each week. (*Role or roles in the project team: Coding, customer relations and design)*
8. No comments. (*Role or roles in the project team:Documentation)*
9. It should be used somehow to divide workload more equitable during course. Not use it only as a note system but take it into the planning process. (*Role or roles in the project team: Project manager)*

6.2.3 The Job Number Log

Question 1: How useful the Job Number was?



Question 2: What were the main benefits in the use of Job Number Log for you and the project team?

Actual answers:

1. For me, no benefits since I did not code anything from scratch. I concentrated fixing the bugs others had made. (*Role or roles in the project team: Testing, coding and planning)*
2. I did not see any benefits using job number log, at least not in the way I used it during the course. (*Role or roles in the project team: Documentation)*
3. I did not use, because I used PSP only for some very small, non-overlapping jobs. (*Role or roles in the project team: Project manager)*
4. I did not help us at all but I did find it very usefully for planning how much time it will take to read a book, for example to an exam. (*Role or roles in the project team: Project manager and programmer)*

5. It helped to remember to do all jobs which have to be done and it helped to divide all jobs to several people. It was then also easier to watch the progress of the project. Unfortunately I started using logging too late. (*Role or roles in the project team: Project manager)*
6. Job number log is nice in the way that it shows all kinds of statistics from a long period of time. It is easier to plan the needed time for new jobs, when you have old data to back you up. Also the LOC-rates proved to be quite interesting. (*Role or roles in the project team: Programmer)*
7. I could see how much time I have to allocate for reading certain amount of pages or coding couple of simple functions. (*Role or roles in the project team: Coding, customer relations and design)*
8. It helped to plan quite accurately the time needed for similar tasks, in my case documenting. I am certain it would have done the same in coding. (*Role or roles in the project team: Documentation)*
9. I didn´t found it very useful but I think that it was not the cause of the method but my role in the project. (*Role or roles in the project team: Project manager)*

Question 3: How would you improve the Job Number Log and its use during the course?

Actual answers:

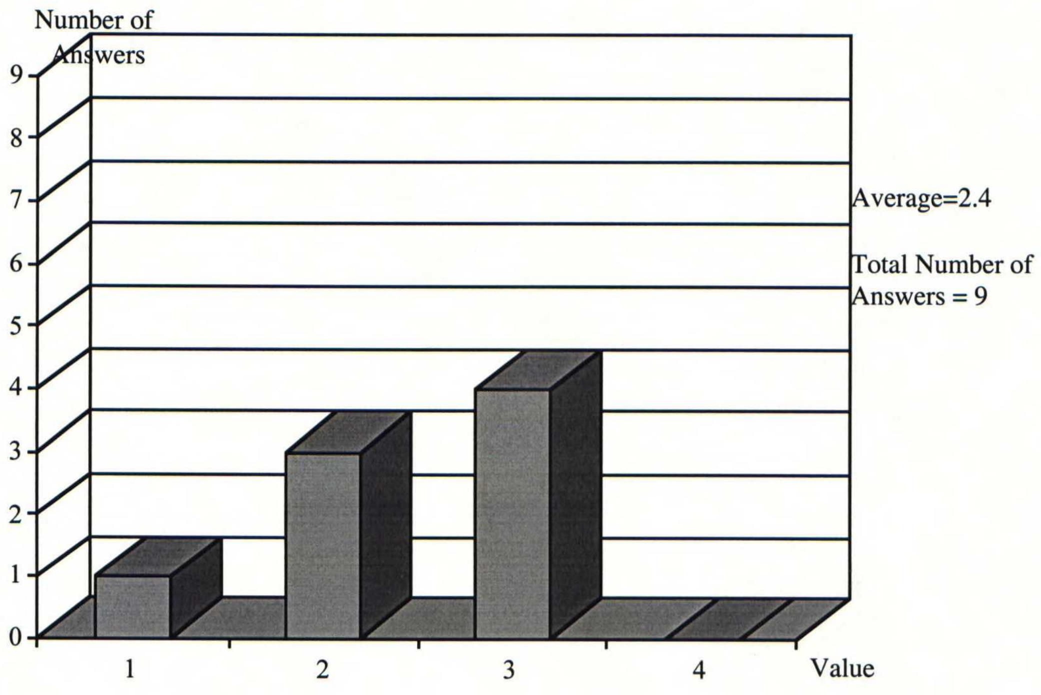
1. Non comments. (*Role or roles in the project team: Testing, coding and planning))*
2. I never understand how do you do good estimations about the amount of work in different tasks, e.g. how do you estimate the amount of lines of code. (*Role or roles in the project team: Documentation)*
3. No comments. (*Role or roles in the project team: Project manager)*
4. No comments. (*Role or roles in the project team: Project manager and programmer)*
5. Using log more would have helped me a lot at least (I find it out now). With Job Log, it is the same thing as with the other logs. The logs should be simpler. Only thing that should be in the logs is the real thing: for example in this case: the real

job description, maybe estimated time but nothing more. Then the time usage should be in time log and copying between different logs just makes more errors, etc.. The summaries and that kind of information should automatically be generate from combining documents. Maybe one other improvement could be that the logs consist not only both your jobs but also the jobs from other people. Then you could also check if everything is taking care of, not only your things. (*Role or roles in the project team: Project manager)*

6. Automation… Most of this data also can already be found on other reports. (*Role or roles in the project team: Programmer)*
7. No comments. (*Role or roles in the project team: Coding, customer relations and design)*
8. It should be defined what type of tasks is reasonable to write done to that log. (*Role or roles in the project team: Documentation)*
9. I think it is fine. (*Role or roles in the project team: Project manager)*

6.2.4 The Whole Time Management Section

Question 1: How useful the whole time management section was?



Question 2: What were the main benefits of whole time management section for you and the project team?

Actual answers:

1. I really can not say. Once you start using some time management system it should apply for all you do. The whole project team had very little time since all was working in real life projects at the same time. (*Role or roles in the project team: Testing, coding and planning)*
2. It was useful for creating reports. (*Role or roles in the project team: Documentation)*
3. See Time Recording Log above. (*Role or roles in the project team: Project manager)*
4. We could better estimate the time we will spend to next phase. (*Role or roles in the project team: Project manager and programmer)*
5. To see how much time it is spend on each category, and to help divide the tasks to group's members so that all things can be done in time. (*Role or roles in the project team: Project manager)*
6. It helps to keep on top of things. Also it gives you an idea of what you spend your productive time on and how (in) efficient you are (LOC-rates etc.). (*Role or roles in the project team: Project manager)*
7. Recording the actual working time quite accurately for the course reporting was the main benefit of the Time Record Log. (*Role or roles in the project team: Coding, customer relations and design)*
8. Total time usage per task, and helped in planning similar tasks. (*Role or roles in the project team: Documentation)*
9. To notice that it is really impossible to do something without breaks and the time spent in breaks is quite big. (*Role or roles in the project team: Project manager)*

Question 3: How would you improve the whole management section and its use during the course?

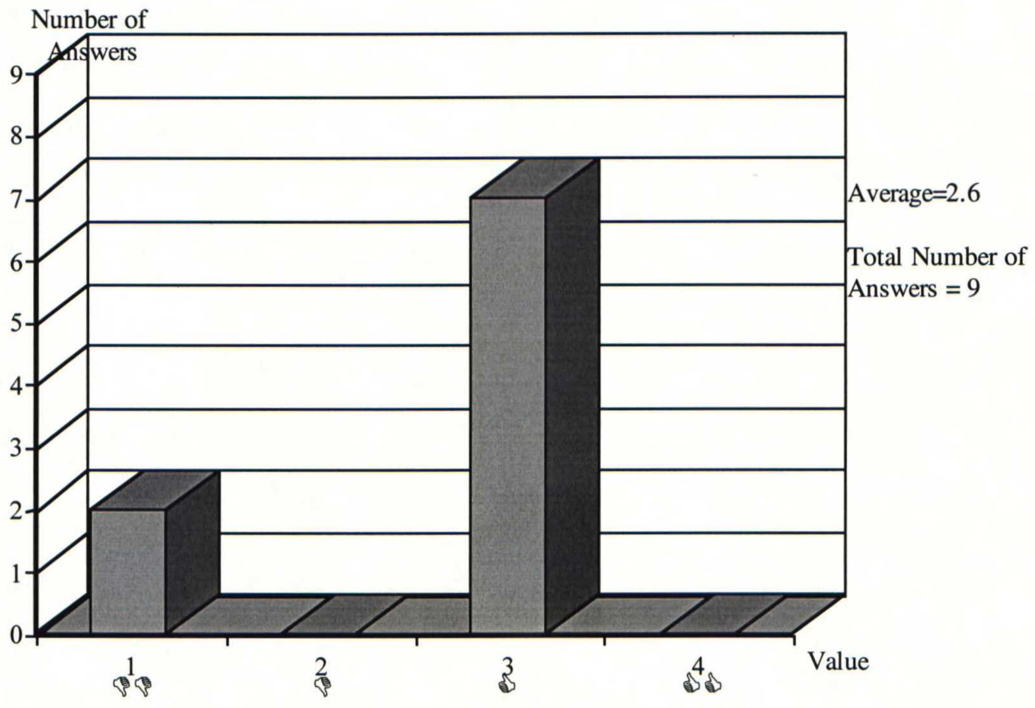
Actual answers:

1. No comments. (*Role or roles in the project team: Testing, coding and planning)*
2. As I said before, it must be done at least semi-automatic. It will not work if you have to have a pad and pen with you all the time. Eventually, you forget to take a pad and pen with you, and everything is just in your memory => not correct times => not correct reports. (*Role or roles in the project team: Documentation)*
3. No need to apply PSP minute tracking to all your daily activities, if your main job is not programming or if you do not aim at overall improvement of your time management skills. (*Role or roles in the project team: Project manager)*
4. Job Number Log could be presented as a tool for studying in generally. (*Role or roles in the project team: Project manager and programmer)*
5. The checking time usage is very useful but the inefficient bureaucracy does no well. The time management should be much simpler. The logs that are hand made should contain no calculated, etc., information, only the real data. All summaries should be automatically generated then by the data. (*Role or roles in the project team: Project manager)*
6. Automation, automation. (*Role or roles in the project team: Programmer)*
7. It helps to keep on top of things. Also it gives you an idea of what you spend your productive time on and how (in) efficient you are (LOC-rates, etc.). (*Role or roles in the project team: Coding, customer relations and design)*
8. The section is fine. It only needs to be used differently (as it is supposed to be used, during office hours). (*Role or roles in the project team:Documentation)*
9. It should be bound to the Software Project course. (*Role or roles in the project team: Project manager)*

6.3 Questions Related to Quality?

6.3.1 The PSP Project Plan Summary?

Question 1: How useful the PSP Project Plan Summary was for your work in the team?



Question 2: What were the main benefits in the use of PSP Plan Summary for you and the project team?

Actual answers:

1. No benefits since the lack of the time. Everybody worked when they could and use of PSP was minimal. (*Role or roles in the project team: Testing, coding and planning)*
2. I did not use this. (*Role or roles in the project team: Documentation)*

3. Seeing the proportions of the time needed for different tasks. It also helped to work more concentrated on the project at hand when you tracked the minutes all the time. (*Role or roles in the project team: Project manager)*
4. We got some information about errors and time estimates for next phase. (*Role or roles in the project team: Project manager and programmer)*
5. We did not have so much time making our quality very high. So we did not make so much testing, etc., so there it was no very exact numbers how many bugs there are, etc. So that kind of summary did not provide any useful information. (*Role or roles in the project team: Project manager)*
6. The use of it was painfully time-consuming, and its benefits to our project remained a mystery. I could not motivate myself for yet another piece of paperwork. *Role or roles in the project team: Programmer)*
7. No comments. *Role or roles in the project team: Coding, customer relations and design)*
8. It had no benefits in documenting. *Role or roles in the project team: Documentation)*
9. I found it quite difficult to apply the LOC-type follow-up. *Role or roles in the project team: Project manager)*

Question 3: How would you improve the PSP Project Plan Summary and its use during the course?

Actual answers:

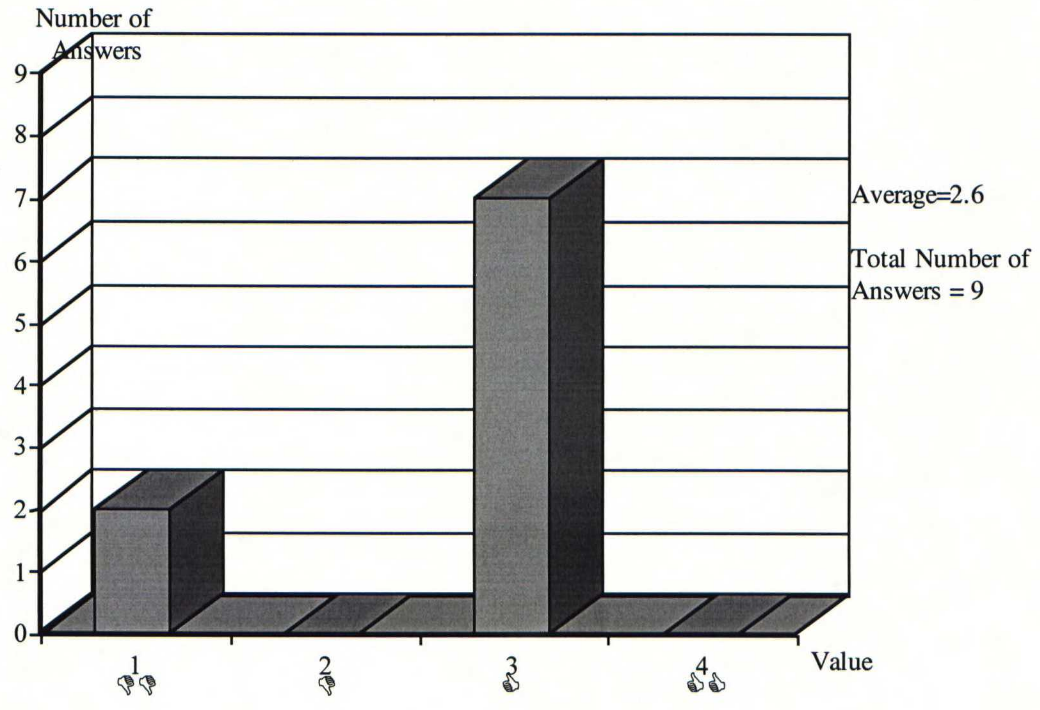
1. No comments. *(Role or roles in the project team: Testing, coding and planning)*
2. No comments. *(Role or roles in the project team: Documentation)*
3. LOC metrics can be misleading. PSP process model is a bit strict, suitable for small projects. The big ones should be broken into small ones of course, but currently there is no support for that in PSP. Better support for spiral development process needed, so that time could be easily accounted for projects after they have been closed, as you often need to make improvements and other maintenance work later. Work needed to integrate software modules (individual PSP projects) not

taken into account. *(Role or roles in the project team: Project manager and programmer)*

4. Fewer things to track, like Yield and A/FR. (*Role or roles in the project team: Project manager)*
5. The summary should better fit the project model used in the course. (*Role or roles in the project team: Project manager)*
6. Automation… Make it easier to fill out. (*Role or roles in the project team: Programmer)*
7. No comments. (*Role or roles in the project team: Coding, customer relations and design)*
8. No suggestion. (*Role or roles in the project team:Documentation)*
9. Maybe module/function based follow-up would be easier to use. (*Role or roles in the project team: Project manager)*

6.3.2 The Defect Recording Log

Question 1: How useful the Defect Recording Log was for your work in the team?



Question 2: What were the main benefits in the use of Defect Recording for you and the project team?

Actual answers:

1. No benefits for the project since nobody really used it. Defects and their correcting were poor in our team: one person took care of the testing, implements did´t record defects, etc.. (*Role or roles in the project team: Testing, coding and planning)*
2. No comments. (*Role or roles in the project team: Documentation)*
3. I did not really use due to small size of my projects. (*Role or roles in the project team: Project manager)*
4. I could more easily track what kinds of errors are the most frequent and maybe avoid them. (*Role or roles in the project team: Project manager and programmer)*
5. Logging errors is good, if every bug is logged then the probability to remember to fix it is much higher. (*Role or roles in the project team: Project manager)*
6. We used a different kind of log only to report the defects, which remained after compilation (and were thus found during system/user testing). Defect recording in this way is of course beneficial to any software project. The data gathered in previous projects gives you an indication what kinds of bugs are the most common and when – and by whom - they are injected. It is a good indicator of software quality, although I disagree with the author of the PSP process on the necessity of reporting every missing semicolon. (*Role or roles in the project team: Programmer)*
7. I could classify the errors I made and I was able to see in which area I had most room for improvement. (*Role or roles in the project team: coding, customer relations and design)*
8. It showed some of my weaknesses in coding. (*Role or roles in the project team: Documentation)*
9. We did not use it as it should. I think that collecting defect data during hole project time is useful and it would have helped our work in later phases of the project. (*Role or roles in the project team: Project manager)*

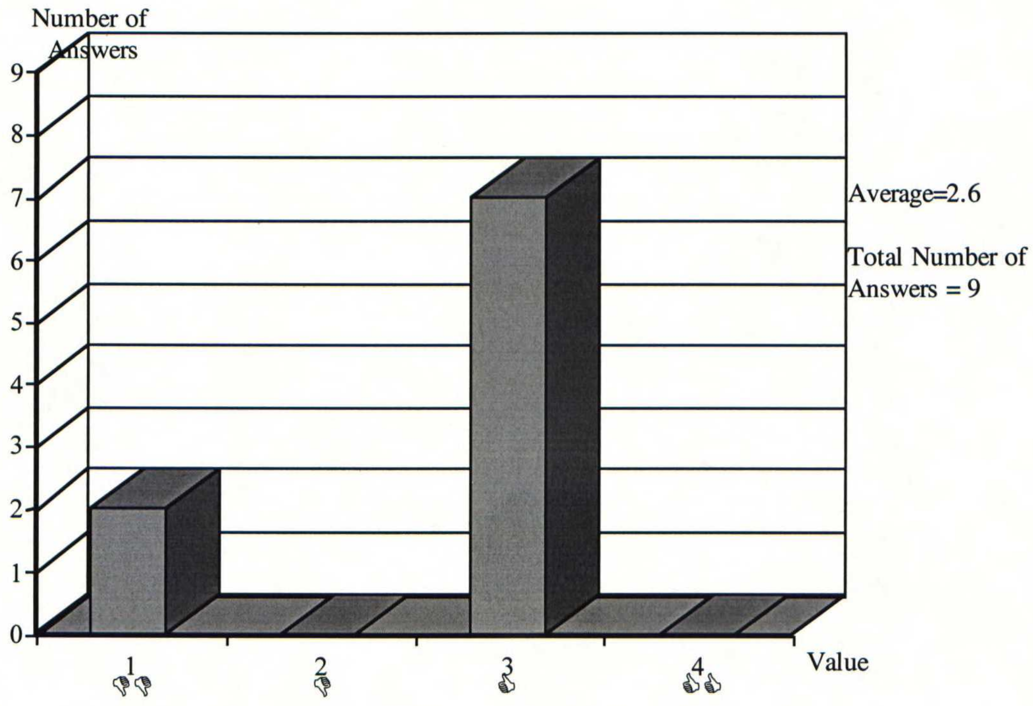
Question 3: How would you improve the Defect Recording Log and its use during the course?

Actual answers:

1. No comments. (*Role or roles in the project team: Testing, coding and planning)*
2. Can it be automatic too? (*Role or roles in the project team: Documentation)*
3. Just filling it in seemed a little pointless because you would need some kind of summaries and a lot of data for it to be useful. (*Role or roles in the project team: Project manager)*
4. The idea is good but I found it somehow too hard to keep. Maybe it could have been better for normal C/C++ coding not for CGI-skripts. (*Role or roles in the project team: Project manager and programmer)*
5. Well, I guess I have said all comments in previous part. (*Role or roles in the project team: Project manager)*
6. With modern powerful compilers and syntax checkers I really cannot see the points of reviewing the code before compile – at least on syntax level. Compilers nowadays catch almost all the syntactic errors and by sensible coding those complicated defects, which escape the compiler can be avoided. I think most programmers do an initial 'reviewing' of the code before they compile, but locating 'defects' like mismatched parentheses (which most commonly are found by the editor you use – say emacs) is unnecessary waste of programming time. (*Role or roles in the project team: Programmer)*
7. No comments. (*Role or roles in the project team: coding, customer relations and design)*
8. There is no point in recording all syntactic errors after you have realized that you make typing errors. Take a course! (*Role or roles in the project team: Documentation)*
9. No comments. (*Role or roles in the project team: Project manager)*

6.3.3 The Personal Checklist

Question 1: How useful the Personal Checklist was for your work in the team?



Question 2: What were the main benefits in the use of Personal Checklist for you and the project team?

Actual answers:

1. I did not use Personal Checklist in the project since I tested functionality only. I have found checklist useful at work where I have used it couple of times. The main benefit is to check the code systematically rather than trying to blindly find out what´s wrong. It brings a systematic way for code checking even before you have tried to compile. (*Role or roles in the project team: Testing, coding and planning)*
2. No comments. (*Role or roles in the project team: Documentation)*
3. I did not use. (*Role or roles in the project team: Project manager)*
4. The idea is good. The problem was that I got almost every time different kind of error. (*Role or roles in the project team: Project manager and programmer)*

5. If there are some bugs, it is easy to first check the easy ones by checking with the list. It surely speeds up the finding bugs. Normally the bugs are harder to find than some typing errors in some line. The bugs are missing functionality or differences between class methods and documentation. I mean bugs come from misunderstandings between people. I cannot see if any checklist would help with that. (*Role or roles in the project team: Project manager*
6. Unfortunately we did not spend much time reviewing the code, but the initial checklist I made proved to be useful during the review. I think, that thorough reviewing of the code requires this kind of list. It is by no means complete: the checklist could be improved significantly by doing a number of code reviews and simultaneously modifying the list to better meet the needs of the project. (*Role or roles in the project team: Programmer)*
7. I didn´t use it. I did not manipulate the project team to use it. Anyway some kind of check list is needed to follow the status of the defects. Again, I used this very little. In conjunction with the Defect Log it revealed my biggest weaknesses and helped me to improve my methods and skills. (*Role or roles in the project team: Coding, customer relations and design)*
8. Again, I used this very little. In conjunction with the Defect Log it revealed my biggest weaknesses and helped me to improve my methods and skills. (*Role or roles in the project team: Documentation)*
9. No comments. (*Role or roles in the project team: Project manager)*

Question 3: How would you improve the Personal Checklist and its use during the course?

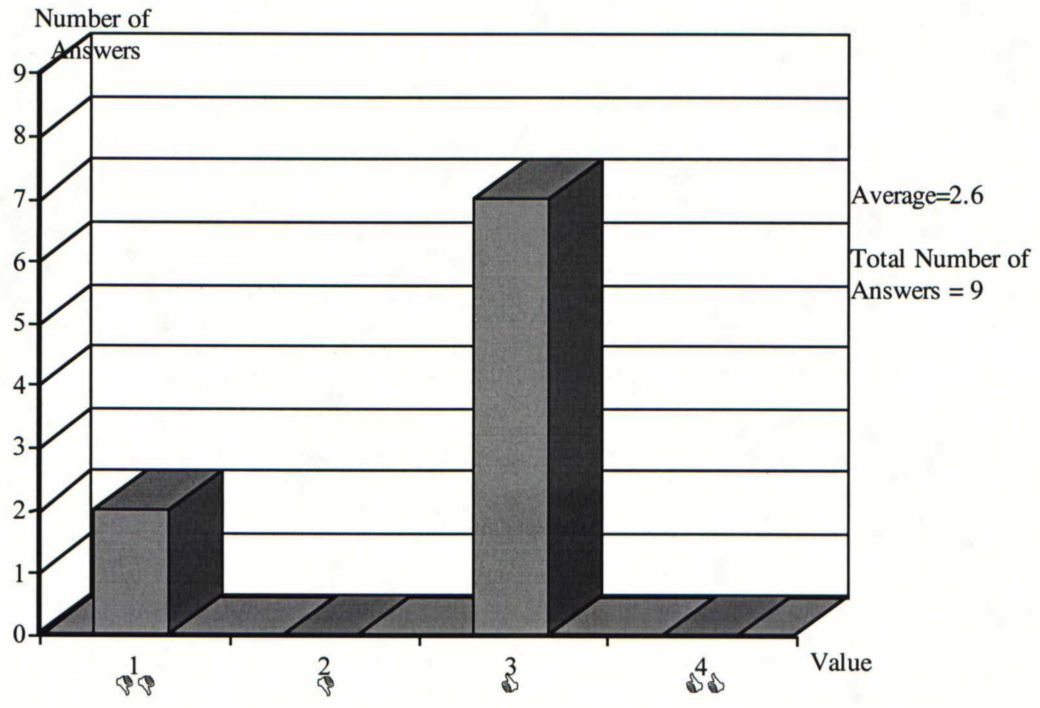
Actual answers:

1. No comments. (*Role or roles in the project team: Testing, coding and planning)*
2. No comments. (*Role or roles in the project team: Documentation)*
3. No comments. (*Role or roles in the project team: Project manager)*
4. No suggestions. (*Role or roles in the project team: Project manager and programmer)*

5. Maybe checklists are good for people who are making their first programs. I guess the main issue is the conversation skills between people. So the main concern should be in defining the strict coding conventions and then working with people so you know what people mean by saying something. (*Role or roles in the project team: Project manager and programmer)*
6. I think it is as good as one makes it, so it is OK as it is. (*Role or roles in the project team: Programmer)*
7. No comments. (*Role or roles in the project team: Coding, customer relations and design)*
8. My checklist was quite small, but I guess that when used in several projects it also becomes better. (*Role or roles in the project team: Documentation)*
9. I do not see what benefits I get when I use those numeric codes in the type column, otherwise the table is OK. (*Role or roles in the project team: Project manager)*

6.3.4 The Whole Quality Section

Question 1: How useful the whole quality section was?



Question 2: What were the main benefits of whole quality section for you and the project team?

1. It is hard to say. It is more in the conceptual level to realize that quality means many things and that there are several ways to make the quality level better. (*Role or roles in the project team: Testing, coding and planning)*
2. No comments. (*Role or roles in the project team: Documentation)*
3. No comments. (*Role or roles in the project team: Project manager)*
4. It was very hard to track quality changes during work. We did most of the developing work in the first coding phase in a two weeks period. We did not have any time to use PSP then. We did not develop enough code so that the results could be trusted. This kind of work would be better on a program course like TiTe B. (*Role or roles in the project team: Project manager and programmer)*
5. It helped to remember the quality. It made me think about it, how to keep quality in some level and how to improve it. (*Role or roles in the project team: Project manager)*
6. It gave us an idea of how software quality can be maintained during the whole project. *Role or roles in the project team: Programmer)*
7. No major benefits. *Role or roles in the project team: Coding, customer relations and design)*
8. It helped me to find my weak areas and methods. *Role or roles in the project team: Documentation)*
9. It is not very rousing to adapt it only from the project manager point of view. *Role or roles in the project team: Project manager)*

Question 3: How would you improve the whole quality section and its use during the course?

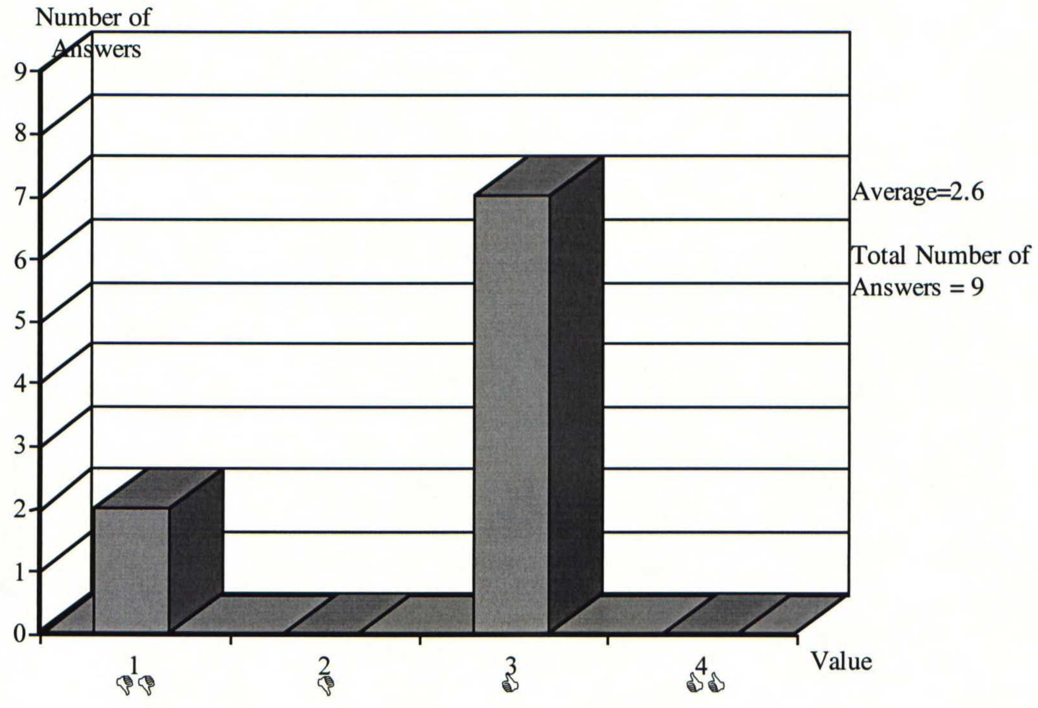
Actual answers:

1. No comments. (*Role or roles in the project team: Testing, coding and planning)*
2. No comments. (*Role or roles in the project team: Documentation)*
3. No comments. (*Role or roles in the project team: Project manager)*

4. Other courses where this could be used. (*Role or roles in the project team: Project manager and programmer)*
5. The quality is more than just bugs in code. The quality is in the eye of the customer. It should be calculated somehow. I do not know how. (*Role or roles in the project team: Project manager)*
6. The role of the Project Plan Summary should be clarified. Also the senseless recompile code reviewing should be changed to recompile code functionality review and postcompile defect reporting. (*Role or roles in the project team: Programmer)*
7. No comments. (*Role or roles in the project team: Coding, customer relations and design)*
8. It might be even more useful if I had done more actual coding. (*Role or roles in the project team: Documentation)*
9. It would be stated in the requirements of the first milestone that project should have clear method to save and follow defects generated and removed during hole project time. (*Role or roles in the project team: Project manager)*

6.4 Questions Related to PSP Course Overall

Question 1: How useful the whole course was, if considering your skills in SW development?



Question 2: What were the main benefits of the course for you?

Actual answers:

1. The main benefit was in the way of thinking: Fast coding (not meaning on the fly) and fewer mistakes by systematic and disciplined way of working. The course introduced a systematic tool for developing software. There were also some good facts available that made it clear why it is important the plan everything you do and do everything in certain way. (*Role or roles in the project team: Testing, coding and planning)*
2. You get a picture of time usage. (*Role or roles in the project team: Documentation)*
3. Made me think about these things, specially time management, even if I am critical towards PSP in its current form. (*Role or roles in the project team: Project manager)*

4. I realized the importance of planning code in advance and use the code review. (*Role or roles in the project team: Project manager and programmer)*
5. The course did help to understand the different aspects of the software process. It gave some very good ideas how to improve the process. It also reminds and the software process is not just children's play with computers. SW process is complicated thing and if good quality is wanted, it needs some resources and very good working methods. PSP can give some ideas to help to find the best working methods for the team. (*Role or roles in the project team: Project manager)*
6. My time management improved significantly. Also I have noticed some improvement in the quality of software I produce. (*Role or roles in the project team: Programmer)*
7. Information about different aspects that should be considered during software development. (*Role or roles in the project team: Coding, customer relations and desing)*
8. I understood the methods how I can observe my methods and processes. I learned to estimate the time that a given task takes based on earlier similar tasks. (*Role or roles in the project team: Documentation)*
9. It is to see alternative way to do following of the project. (*Role or roles in the project team: Project manager)*

Question 3: What were the main aspects that you would like to change in the course?

Actual answers:

1. The seminars were good. Things should be automated enough to avoid certain frustration in filling up all the paperwork (templates, etc.). Some assignments like Job Number Log did´t fit in to this course. (*Role or roles in the project team: Testing, coding and planning)*
2. No comments. (*Role or roles in the project team: Documentation)*
3. Made me think about these things, especially time management, even if I´m critical towards PSP in its current form. (*Role or roles in the project team: Project manager)*
4. PSP could be integrated to a different kind of course and results could be sent every one or two weeks. (*Role or roles in the project team: Project manager and programmer*
5. I guess the main problem with the course is the material in PSP book does not fit very well in the process used by Software Project Course. Also the logging procedure is made too complex and time consuming so it easily is forgotten. So what I would like to see changed is the logs and summaries are developed further. Logs should be easier to fill and summaries should be automatically generated. The process model also should be the same as used in the Software Project course. (*Role or roles in the project team: Project manager)*
6. The reporting should be less tedious. Overall it is good and well suited for multiple small software projects. (*Role or roles in the project team: Programmer)*
7. It should be adapted better for the SW Development course or kept as totally separate course. PSP could work better with for example TiTe B course. (*Role or roles in the project team: Coding, customer relations and desing)*
8. The way it is used. Take a shorter period of time, say a month or so, and follow the time usage every day for every task, not just for one course. Following just one course gives data, but it is not that kind of data Humprey probably had in mind…(*Role or roles in the project team: Documentation)*

9. This should be a part of the Software Project course so then everyone who works for the project will commit to same rules. (*Role or roles in the project team: Project manager)*

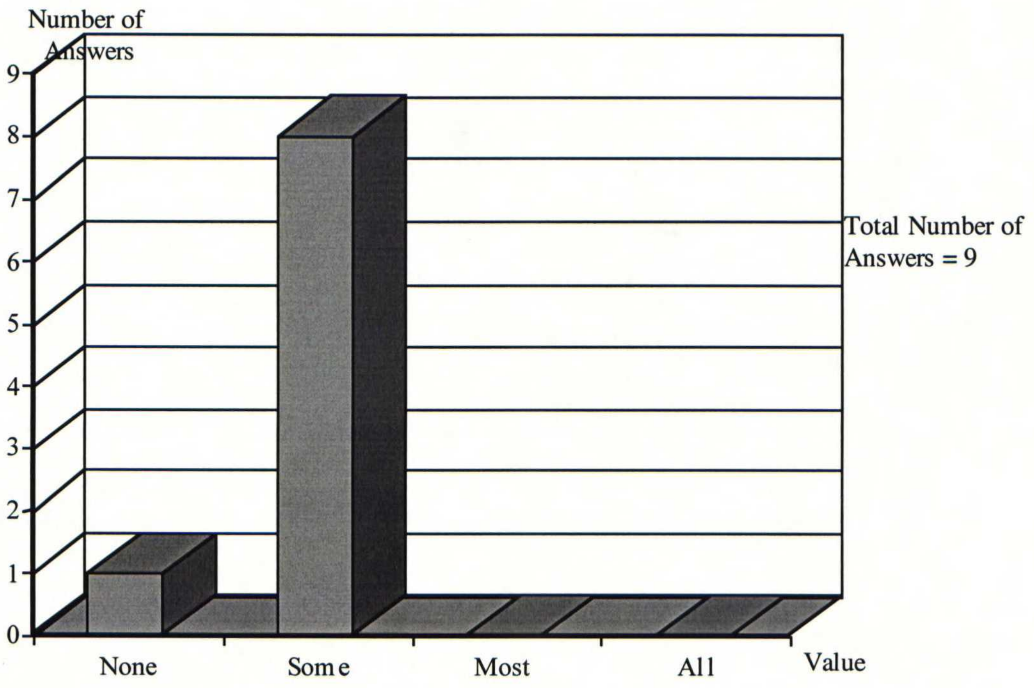
Question 4: How useful did you find presentations of the PSP topics by fellow students? How would you improve introductions of the topics?

Actual answers:

1. The current way is good and useful. (*Role or roles in the project team: Testing, coding and planning)*
2. I personally think that filling out the forms is waste of time. As I said before, pretty much everything must be automated in some level. The updating of the log must be so easy, that it does not cause any overhead to your work. (Not a answer to the question itself ☺). Basically, the seminar kind approach is pretty good. I can't figure out any better way to actually introduce the topics. (*Role or roles in the project team: Documentation)*
3. Need an automatic tool for recording data and making reports! Doing it by hand is tedious and obscures the real point. More flexible process model would be needed, but is it PSP any more? *(Role or roles in the project team: Project manager)*
4. In generally, they were too kind of direct reading from a book. They should have been shorter and maybe more discussion around the topic. *(Role or roles in the project team: Project manager and programmer)*
5. I guess, I will use the time logging by somer way to see how my time really goes. And that job logging too. I think they were the best ideas in PSP. The others are also good things to remember but I do not know if this solution is the best. I think they have to have more work to be done to develop them better. *(Role or roles in the project team: Project manager)*
6. It was nice to listen to other students′ presentations and opinions on the PSP topics. I think it si good as it. *(Role or roles in the project team: Programmer)*

7. The reporting should be less hard. Overall it is good and well suited for multiple small software projects. It was nice to listen to other students' presentations and opinions on the PSP topics. I think it is good as it is. I think I will keep using those parts of the PSP. I liked mostly the time management and some quality related things. It was nice.
   The reporting could be improved a lot to make it more user friendly. (*Role or roles in the project team: Coding, customer relations and desing)*
8. Mostly they were OK. Everybody can read the book, so people should try to pick up the essential things and represent them. (*Role or roles in the project team: Documentation)*
9. They were very useful! It helped to understand the material better with less actual reading. I have no suggestions to our presentations. The presentations were top class! (*Role or roles in the project team: Project manager)*

Question 5: How much you expect you will use the working practices, which you applied during the PSP course, in the future? Select one of the following alternatives: None /Some/Most/All.



Question 6: Any other suggestions for improving the course?

1. No comments. (*Role or roles in the project team: Testing, coding and planning)*
2. It was good that the course was related to the specific course. It should be integrated into the core of the course more tight. I think that I made myself clear about the automation of the process, as much as possible. (*Role or roles in the project team: Documentation)*
3. No comments. (*Role or roles in the project team: Project manager)*
4. No comments. (*Role or roles in the project team: Project manager and programmer)*
5. I think this course is quite fine. (*Role or roles in the project team: Project manager)*
6. It was nice. The reporting could be improved a lot to make it more user friendly. (*Role or roles in the project team: Programmer)*
7. No comments. (*Role or roles in the project team: Coding, customer relations and design)*
8. Probably continue tracking time of relatively small projects and wait for an enlightenment of how to get real benefit out of it. *(Role or roles in the project team: Documentation)*
9. This may even turn to Most if it helps to improve my methods at my work. No comments. (*Role or roles in the project team: Project manager)*

6.5 Summary of the Feedback from the Pilot PSP Course at Helsinki University of Technology

Students´ comments about the pilot PSP course was very encouraging. It seems that some critical goals of the PSP course were obtained. The students started to think of the significance of time management, planning, and software development. *"I understood the methods how I can observe my methods and processes. I learned to estimate the time that given task takes based on earlier similar tasks."*

The time management seems to be relevant for student's project work. The Time Recording Log supplied for PSP provides an ideal template for recording time spent in the project. The time data gives better understanding of how much time each task takes, and how much non-development time (for example, meetings, reports, interruptions, etc.) there really is. *"It was terrible to follow in what kind of issues my time actually goes. I notice that it is really impossible to do something without breaks and the time spent in breaks is quite big."*

Gathering defect data seems to be very difficult to students. Only some student used the Defect Recording Log and the Personal Checklist. *"We didn't use it as it should. I think that collection defect data during whole project time is useful, and it would have helped our work in later phases of the project. The idea is good but I found it somehow too hard to keep. Maybe it could have been better for normal C/C++ coding not for CGI-skripts."* The students recommended to use the quality section forms, The Defect Recording Log and Personal Checklist, at early freshman's courses (for example, Tik-76.020 Basic Course in Programming L1 or Tik-76.021 Basic Course in Programming L2)

The most critical improvement areas for future PSP courses seem to be:

1. Automatization of data collection and analysis: The collection of time and defect data shall be simplified and automated. A limited number of tools are available via www, but all of those have limitations. HUT should probably develop own tools based on other universities' experiences. That could be for example be one topic for SW Project Course.

2. Integration with other course or courses: The PSP methods should more tightly integrate with another computer science course or courses. It might be even reasonable to consider how the PSP principles could be integrated throughout the CS curriculum, because learning good working practices take a lot of time. The full PSP activities could be integrated to freshman course, for example, Tik-76.020 Basic Course in Programming. The time management activities could be integrated to Tik-76.115 Software Project course.

8. CONCLUSIONS

The PSP is a very promising tool for the student and the engineer who is willing to work with PSP tables, forms, and scripts. The text *Introduction to Personal Software Process* [Hump97b] serves as an ideal companion text for the computer science first year courses. Introduction of this new technology is not trivial and requires extensive planning, data collection and monitoring. The textbook is self-contained, experienced students could use the textbook to help them learn the PSP on their own, but most students need the structure and support of a formal training course to complete the training. Learning the PSP on your own is a major effort. Sharing lessons learned with other students that are using the PSP on the project would certainly help.

PSP training requires a lot of student´s time. The entire approach of collecting data, making process changes based on that data, and then measuring the effects of those changes takes considerable time. The student must invests this time to see the benefit of discipline, and must be prepared for the hard work of changing the way she/he works. Patience and commitment are required in using the PSP. However the PSP concept needs to be integrated throughout the computer science curriculum.

Substantial, integrated automated support for both collection and analysis stages in PSP are essential, if high data quality is desired. Automation of the analysis stage is not a new idea to the PSP, and many spreadsheets and the database tools have already been made. Automated and integrated support both collection and analysis stages in the PSP is a challenging goal for future research on personal software process improvement

The teacher of the PSP course shall have enough experience of using PSP. After you have personally used the PSP, you have background to explain the PSP to the students and support their studies.

REFERENCES

[AlVa98] Alho, Kari and Vanhanen, Jari. Tik-76.115 Software Project (5 cr.) course. 1998 Helsinki University of Technology. URL:http://www.mordor.cs.hut.fi/tik-76.115/oht.htm

[BaFu95] Bandinelli, Sergio, Fugetta, Alfonso , Lavazza, Luigi, Loi, Maurizio, and Picco, Gian Pietro. Modeling and Improving an Industrial Software Process. IEEE Transactions on Software Engineering. Vol. 21, No. 5, May 1995. pp. 440-454

[BaRo88] Basili, Victor R. and Rombach, H. Dieter. The TAME Project: Towards Improvement-Oriented Software Environments. IEEE Transactions on Software Engineering. Vol. 14. No. 6. June, 1988.

[BaCR94] Basili, Victor R., Caldiera, Gianluigi and Rombach, H. Dieter. The Experience Factory. Encyclopedia of Software Engineering. Wiley 1994.

[BaMc95] Basili, Victor R. and Turner, A.J. Tutorial: The Experience Factory: How to Build and Run One. 17th International Conference on Software Engineering. USA: Seattle, Washington. April 24, 1995.

[EmSM96] Emam, Khaled El, Shostak, Barry and Madjavji, Nazim. Implementing Concepts from the Personal Software Process in an Industrial Setting. Proceeding of the 4th International Conference on the Software Process, pages 117-130, IEEE CS Press, 1996.

[FeHu93] Feiler, P.H., and Humphrey, W. S.. Software Process Development and Enactment: Consepts and Definitions. In Proceedings of the second International Conference on the Software Process, IEEE Computer Society Press, 1993, pp. 28-40.

[FeHu97] Ferguson, Pat, Humphrey, Watts S., Khajenoori, Soheil, Macke, Susan and Matvya, Annette. Result of Applying the Personal Software Process. 0018-9162/97. 1997 IEEE.

[HaOv97] Hayes, Will and Over, James W. The Personal Software Process (PSP): An Empirical Study of the Impact of PSP on Individual Engineers. Technical Report. CMU/SEI-97-TR-001. Pittsburgh, PA, USA: The Software Engineering Institute, 1997

[HeGo96] Herbsleb, James D. and Goldenson, Dennis R. A Systematic Survey of CMM Experience and Results. CMU/SEI-0270-5257/96. Pittsburgh, PA, USA: The Software Engineering Institute, 1996 IEEE.

[Hump89] Humphrey, Watts S. Managing the Software Process. Reading, USA: Addison Wesley, 1989. 494 p. ISBN 1-55937-067

[Hump94a] Humphrey, Watts S. The Personal Software Process. In Software Process Newsletter, IEEE Computer Society TCSE, No. 1, p. 1-3, September 1994.

[Hump94b] Humphrey, Watts S. The Personal Software Process Overview, Practice, and Results. 6 p. S.P.I. Forum, 1994.

[Hump94c] Humphrey, Watts S. The Personal Process in Software Engineering. The Third International Conference on the Software Process, Reston, Virginia, October 10-11, 1994, pp. 69-77.

[Hump94d] Humphrey, Watts S. A Personal Commitment to Software Quality. American Programmer, December 1994

[Hump95] Humphrey, Watts S. A Discipline for Software Engineering. Reading, USA: Addison Wesley, 1995. 789 p. ISBN 0-201-54610-8.

[Hump96a] Humphrey, Watts S. Using a Defined and Measured Personal Software Process. In IEEE Software, pages 77-88, May 1996.

[Hump97a] Humphrey, Watts S. Managing Technical People: Innovation, Teamwork, and the Software Process. Reading, USA: Addison Wesley, 1997. 326 p. ISBN 0-201-54597-7.

[Hump97b] Humphrey, Watts S. Introduction to the Personal Software Process. Reading, USA: Addison Wesley, 1997. 278 p. ISBN 0-201-54809-7.

[Hump97c] Humphrey, Watts S. The Personal Software Process (PSP)^{SM} Tutorial. The Second Annual: European Software Engineering Process Group Conference 1997. Amsterdam. 16-19th June 1997.

[Hump97d] Humphrey, Watts S. Instructor´s Guide and Assignment Kits for Introduction to the Personal Software Process. Addison Wesley, 1997. ISBN 0-201-17356-5.

[Hump98a] Humphrey, Watt S., Article: Three Dimensions of Process Improvement. Part I: Process Maturity. Crosstalk - The Journal of Defense Software Engineering. February 1998.

[Hump98b] Humphrey, Watt S., Article: Three Dimensions of Process Improvement. Part II: The Personal Process. Crosstalk - The Journal of Defense Software Engineering. February 1998

Hump98c] Humphrey, Watt S., Article: Three Dimensions of Process Improvement. Part III: The Team Process. Crosstalk - The Journal of Defense Software Engineering. April 1998

[Hump98d] Humphrey, Watt S., Why Don´t They Practice What We Preach? 1998 Carnegie Mellon University. URL: http://www.sei.cmu.edu/publications/articles/sources/ practice.preach/index.html. Last Modifield: 29 May 1998

[Hump98e] Humphrey, Watt S., The Team Software Process (TSP). The Second Annual: European Software Engineering Process Group Conference 1998. London June 1998.

[IEEE90] IEEE Std 610.12-1990, Standard Glossary of Software Engineering Terminology. IEEE, New York, USA 1990.

[Khaj94] Khajenoori, Soheil. Process-Oriented Software Education. IEEE Software, pages 99-101, November 1994.

[KeBr96] Kellner, Marc I., Briand, Loic and Over, James W., A Method for Designing, Defining and Evolving Software Processes. 0-8186-7719-8/96. 1996 IEEE. PP. 37-48.

[MaKN96] Macke, Susan, Khajenoori, Soheil, New, Jeff, Coxon, Jed and Rockwell, Ron. Personal Software Process At Motorola Paging Products Group. In Proceedings of the Software Engineering Process Group Conference, 1996.

[Matv96] Matvya, Annette. Industrial Strength PSP at Union switch and Signal Inc. In Proceedings of the Software Engineering Process Group Conference, 1996.

[McFe96] McFeeley, Bob. IDEAL ^{SM}: A User´s Guide for Software Process Improvement. Handbook CMU/SEI-96-HB-001. Pittsburgh, PA, USA: Software Engineering Institute, 1996.

[PaulCC93] Paulk, M.C. et. al. Capability Maturity Model for Software, Version 1.1. Technical Report CMU/SEI-93- TR-24. Pittsburgh, PA, USA: Software Engineering Institute, 1993.

[Paul94] Paulk, Mark C., Weber, Charles V., Curtis, Bill and Chrissis, Mary Beth. The Capability Maturity Model: Guidelines for Improving the Software Process. ISBN 0-201-54664-7. Addison-Wesley Publishing Company. Reading. MA 1994.

[PaKG95] Paulk, Mark C., Konrad, Michael D. and Garcia, Suzanne M.. CMM Versus SPICE Architectures. Published in Software Process Newletter, IEEE Computer Society Technical Council on Software Engineering, No. 3, Spring 1995, pp. 7-11.

[Paul96] Paulk, Mark C. Effektive CMM-Based Process Improvement. Proceedings of the 6th Conference on Software Quality, Ottawa, Canada. 28-31 October 1996, PP. 226-237.

[Proc98] Personal Software Process (PSP). Process Term.
URL: http://www.sei.cmu.edu/psp/ProcessTerms.htm
Last Modified: 30 April 1998.

[Rams96] Ramsey, Margaret A., Experience Teaching the PSP in Academia and Industry. In proceedings of the Software Engineering Process Group Conference, 1996.

[Roy96] Roy, Daniel M., The Personal Software Process: An ¨ego-centered¨ improvement paradigm. In proceedings of the Software Engineering Process Group Conference, 1996

[Smit98] Home Page for Smal Software Project Development Courses by M.R.Smith.
URL:http.//www.enel.ucalcary.ca/People/Smith/619.94

[ToHi97] Towhidnejad, Massood and Hilburn, Thomas. Integrating the Personal Software Process (PSP) across the Undergraduate Curriculum. 1997 ASEE/IEEE Frontiers in Education Conference.

[Will97] Williams, Laurie A. Adjusting the Instruction of the Personal Software Process to Improve Student Participation. In proceedings of the Software Engineering Process Group Conference, 1996.

APPENDIX 1: THE GRADUATE PSP COURSE [HUMP98D]

Objectives	1. To improve a software engineers performance. 2. To provide an understanding of what processes are, how processes work, and how a personal process can help the students do better work. 3. To help engineers measure the quality of their programs and determine how to consistently produce high-quality programs. 4. To provide engineers with measures of their personal performance and benchmarks against which to judge their improvement. 5. To show engineers how to make accurate plans.
Prerequisites	1. Fluency in at least one programming language. 2. A basic understanding of software design. 3. Familiarity with basic mathematics through calculus. 4. An appreciation of statistical principles is helpful not essential. 5. An awareness of formal methods is also helpful but not essential.
Course Structure	1. The basic PSP course has 15 lectures of 50 minutes each. 2. An additional laboratory per week is generally helpful in assisting the students to complete their work properly and on time. 3. There are 10 programming assignments that average about 100 LOC each, although program 10 is generally a little larger. 4. Five report assignment require the students to analyze personal data. 5. Students generally take 4 to 6 hours to complete each programming exercise, on to two hours each for reports 1, 2, and 3, and about 6 to 8 hours each for reports 4 and 5.
Course Support	1. The course is fully described in Humphrey´s textbook: *A Discipline for Software Engineering,* Addison Wesley, 1995. 2. An instructor´s quide describes lecture objectives, suggests grading criteria, comments on the assignments, and provides lecture overheads. 3. An instructor's diskette includes spreadsheets for analyzing student data and lecture overheads. 4. A support diskette provides data analysis and calculation support for the students.

APPENDIX 2. THE UNDERGRADUATE PSP COURSE STRUCTURE [HUMP98D]

Objectives	1. To provide beginning software engineering and computer science students with an appreciation of personal software engineering disciplines. 2. To expose students to planning and time management methods. 3. To provide working knowledge of basic software quality practices. 4. To provide a disciplined learning framework that will counter the common hacker ethic that many new students develop.
Prerequisites	1. The basic requirements for the CS1 and CS2 courses.
Course Structure	1. The freshman PSP course is a companion to any other beginning two-semester pair of software or computer science courses that include writing 8 or more small programs. 2. The students do the regular course work, augmented with the PSP materials. 3. The PSP materials specify how the students are to do their regular work using disciplined methods. 4. The PSP material takes about 6 lecture hours spread over the two semesters. 5. It is suggested that these added lectures be given at the beginning of each of the two semesters.
Course Support	1. The course is fully described in Humphrey′s textbook: *Introduction to the Personal Software Process*, Addison Wesley, 1997. 2. An instructor's guide describes the lecture objectives, suggests grading criteria, comments on the assignment, provides a set of lecture overheads, and gives suggested quizzes and quiz answers. 3. An instructor's diskette includes spreadsheets for analyzing student data, lecture overheads, and quizzes. A support diskette provides student data analysis and calculation support.

APPENDIX 3. THE PLANNED TSP UNDERGRADUATE OR GRADUATE TEAM-WORKING COURSE STUCTURE [HUMP98D]

Objectives	1. To provide computer science or software engineering students with practical experience working in a team development environment. 2. To show such students how to plan and manage their own work in team environment and under typical management conditions. 3. To show students how to apply the knowledge they have gained in a practical engineering setting. 4. To demonstrate the teamwork benefits of following disciplined individual methods. 5. To learn how and when to ask for and give team support and assistance.
Prerequisites	1. The course would be designed to support either a full one or two-semester project course or a shorter team project in another course. 2. The ability and an interest in working cooperatively with other students on a demanding project. 3. Basic competency in designing and developing small to moderate-sized programs.
Anticipated Course Structure	1. The course would be designed to support either a full one or two-semester project course or a shorter team project in another course. 2. The students would typically work in 4 to 6-person teams. 3. The team process would start with an abbreviated launch workshop where the team would plan their project. 4. The course materials will provide several predefined optional project, each of which could be completed in either the one- or two-semester course format.
Anticipated Course Support	1. A textbook, including an academic version of the industrial-scale TSP process and exercise specifications. 2. An instructor's guide to show a PSP-qualified faculty member or graduate student how to act as the team manager and coach. 3. Faculty and student data gathering and analysis support.

APPENDIX 4: THE PSP PROJECT PLAN SUMMARY INSTRUCTIONS [HUMP95A]

Purpose	This form holds the estimated and actual project data in a convenient and readily retrievable form.
Header	Enter the following: • Your name and today's date • The program name and number • The instructor's name • The language you will use to write the program
Minutes/LOC	Prior to development • Enter the Minutes/LOC planned for this project. Use the to Date rate from the most recent program in the Job Number Log or the most recent Project Plan Summary. After development • Divide the total development time by the actual program size to get the actual and To Dates Minutes/LOC. • For example, if the project took 196 minutes and you produced 29 LOC, the Minutes/LOC would be 196/29= 6.76.
LOC/Hour	Prior to development • Calculate the LOC per hour planned for this program by dividing 60 by the Plan Minutes/LOC After development • For Actual and To Date LOC/Hour, divide 60 by the Actual To Date Minutes/LOC • For Actual Minutes/LOC of 6.76, Actual LOC/Hour are 60/6.76= 8.88.
Defect/KLOC	Prior to development • Find the defects/KLOC To Date on the most recent previous program. • Use this as the Plan Defects/KLOC for this project. After development • Calculate the defects/KLOC actual and To Date for this program. • For Actual, multiply the total actual defects by 1000 and divide by the Actual Total New & Changed LOC. • Make a similar calculation for To Date. • With 17 defects to date and 153 Total New & Changed LOC, defects/KLOC To Date = 1000*17/153 = 111.1.

Yield	Calculate the plan, actual, and to date yield. Yield = 100*(defects removed before compile)/(defects injected before compile), so with 5 injected and 4 found, yield=100*4/5=80.0 %.
A/FR	Calculate the plan, actual, and To Date A/FR • For actual, for example, take the ratio of the actual code review time and divide by the sum of the actual compile and test times. • For review time of 29 minutes, compile time of 5 minutes, and test time of 10 minutes, A/FR = 29/(5+10)=1.93.
Program Size - (LOC)	Prior to development • Enter under plan the estimated Total, Maximum, and Minimum New & Changed LOC. After development • Count and enter the Actual New & Changed LOC. • For To Date, add Actual New & Changed LOC to the To Date New & Changed
Time in Phase - Plan	For total development time, multiply Total New & Changed LOC by Minutes/LOC: For Maximum time, multiply the Maximum size by Minutes/LOC. For Minimum time, multiply the Minimum size by Minutes/LOC. From the Project Plan Summary for the most recent program, find the to Date % values for each phase. Using the To Date % from the previous program, calculate the plan time for each phase.
Time in Phase - Actual	At job completion, enter the actual time in minutes spent in each development phase. Get these data from the time log.
Time in Phase - To Date	For each phase, enter the sum of actual time and to To Date time from the most recent previous program.
Time in Phase- To Date %	For each phase, enter 100 times the To Date time for that phase divided by the Total To Date time.
Defects Injected - Plan	Before development, estimate the total number of defects to be injected in the program. The value is Plan Defects/KLOC times the Plan Total New & Changed LOC for this program divided by 1000. For example, with a Plan Defects/KLOC of 75.9 and a Plan New & Changed LOC of 75, Plan Total defects = 75.9*75/1000=5.69, so use 6. Before development, estimate the defects injected by phase using the estimate total defects and the To Date % defect injected distribution from the previous program.
Defects Injected - Actual	After development, find and enter the actual number of defects injected in each phase.

Defects Injected - To Date	For each phase, enter the sum of the actual defects and the To Date defects from the most recent program.
Defects Injected - To Date %	For each phase, enter 100 times the To Date defects for that phase divided by the total To Date defects.
Defects Injected - Defects/hour	Calculate the defects injected per hour for design and code. For design, for example multiply 60 times the design defects To Date and divide by the design time To Date = 60*5/195 = 1.54 defects/hour.
Defects Removed - Plan	In the total row, enter the estimated total defects. Using the To Date % values from the most recent program, calculate the plan defects removed for each phase.
Defects Removed - Actual	After development, find and enter the actual number of defects removed in each phase.
Defects Removed - To Date	For each phase, enter the sum of the actual defects and the To Date defects from the most recent program.
Defects Removed - To Date %	For each phase, enter 100 times the To Date defects for that phase divided by the total To Date defects.
Defects Removed - Defects/hour	Calculate the defects removed per hour for code review, compile, and test. For test, for example, multiply 60 times the test defects To Date and divide by the test time To Date = 60*6/279 = 1.29 defects/hour.

APPENDIX 5: FEEDBACK QUESTIONNAIRE

Feedback Questionnaire - Tik-76.161 Individual Studies - Personal Software Process (3 cr) P

Introduction

The purpose of the questionnaire is to get feedback for the improvement of this course Tik-76.161 Individual Studies - Personal Software Process (PSP) (3 cr) P. The course applying PSP was arranged first time in Helsinki University of Technology during 97 – 98. Your feedback is vital for getting more out of PSP in the following courses.

Several questions in this questionnaire ask about usefulness using a four level scale. In each questions you should consider following two aspects:

- How well it teaches the purpose and practices of the subject matter?
- How beneficial is it for the actual SW development?

In the last section you are asked to answer more general questions about the course.

Identification

Name:

Role(-s) in the project team:

Questions related to Time Management

Time Recording Log:

How useful the Time Recording Log was in planning and tracking your work in the team?

👎👎	👎	👍	👍👍
☐	☐	☐	☐

What were the main benefits in the use of Time Recording Log for you and the project team?

How would you improve the Time Recording Log and its use during the course?

Weekly Activity Summary:

How useful the Weekly Activity Summary was?

👎👎	👎	👍	👍👍
☐	☐	☐	☐

What were the main benefits in the use of Weekly Activity Summary for you and the project team?

How would you improve the Weekly Activity Summary and its use during the course?

Job Number Log:

How useful the Job Number Log was?

👎👎	👎	👍	👍👍
☐	☐	☐	☐

What were the main benefits in the use of Job Number Log for you and the project team?

How would you improve the Job Number Log and its use during the course?

Whole time management section:

How useful the whole time management section was?

👎👎	👎	👍	👍👍
☐	☐	☐	☐

What were the main benefits of whole time management section for you and the project team?

How would you improve the whole time management section and its use during the course?

Questions related to Quality

PSP Project Plan Summary:

How useful the PSP Project Plan Summary was for your work in the team?

👎👎 👎 👍 👍👍

☐ ☐ ☐ ☐

What were the main benefits in the use of PSP Project Plan Summary for you and the project team?

How would you improve the PSP Project Plan Summary and its use during the course?

Defect Recording Log:

How useful the Defect Recording Log was for your work in the team?

👎👎 👎 👍 👍👍

☐ ☐ ☐ ☐

What were the main benefits in the use of Defect Recording Log for you and the project team?

How would you improve the Defect Recording Log and its use during the course?

Personal Checklist:

How useful the Personal Checklist was for your work in the team?

👎👎	👎	👍	👍👍
☐	☐	☐	☐

What were the main benefits in the use of Personal Checklist for you and the project team?

How would you improve the Personal Checklist and its use during the course?

Whole quality section:

How useful the whole quality section was?

👎👎	👎	👍	👍👍
☐	☐	☐	☐

What were the main benefits of whole quality section for you and the project team?

How would you improve the quality section and its use during the course?

Questions related to Course Overall

How useful the whole course was, if considering your skills in SW development?

👎👎 👎 👍 👍👍

☐ ☐ ☐ ☐

What were the main benefits of the course for you?

What are the main aspects that you would like to change in the course?

How useful did you find presentations of the PSP topics by fellow students? How would you improve introduction of the topics?

How much you expect you will use the working practices, which you applied during PSP course, in the future? Select one of the following alternatives:

None / Some / Most / All

Comments:

Any other suggestions for improving the course?

Thank you! ☺