

AUTOMATED SUPPORT FOR TECHNICAL SKILL ACQUISITION AND  
IMPROVEMENT: AN EVALUATION OF THE LEAP TOOLKIT

A DISSERTATION PROPOSAL SUBMITTED TO THE GRADUATE DIVISION  
OF THE UNIVERSITY OF HAWAI'I IN PARTIAL FULFILLMENT  
OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMMUNICATION AND INFORMATION SCIENCES

By

Carleton A. Moore

Dissertation Committee:

Philip Johnson, Chairperson

James Corbett

Elizabeth Davidson

Marie Iding

Larry Osborne

November 30, 1999

Version 2.0.3

# Abstract

Software developers work too hard and yet do not get enough done. Developing high quality software efficiently and consistently is a very difficult problem. Developers and managers have tried many different solutions to address this problem. Recently their focus has shifted from the software organization to the individual software developer. The Personal Software Process incorporates many of the previous solutions while focusing on the individual software developer.

I combined ideas from prior research on the Personal Software Process, Formal Technical Review and my experiences building automated support for software engineering activities to produce the Leap toolkit. The Leap toolkit is intended to help individuals in their efforts to improve their development capabilities. Since it is a light-weight, flexible, powerful, and private tool, it allows individual developers to gain valuable insight into their own development process. The Leap toolkit also addresses many measurement and data issues involved with recording any software development process.

The main thesis of this work is the Leap toolkit provides a more accurate and effective way for developers to collect and analyze their software engineering data than manual methods. To evaluate this thesis I will investigate three claims: (1) the Leap toolkit prevents many important errors in data collection and analysis; (2) the Leap toolkit supports data collection and analyses that are not amenable to manual enactment; and (3) the Leap toolkit reduces the level of “collection stage” errors. To evaluate the first claim, I will show how the design of the Leap toolkit effectively prevents important classes of errors shown to occur in prior related research. To evaluate the second claim, I will conduct an experiment investigating 14 different quantitative time estimation techniques based upon historical size data to show that the Leap toolkit is capable of complex analyses not possible in manual methods. To evaluate the third claim, I will analyze software developers data and conduct surveys to investigate the level of data collection errors.

# Table of Contents

Abstract . . . . .	ii
List of Tables . . . . .	vi
List of Figures . . . . .	vii
1 Introduction . . . . .	1
1.1 Why is Quality Software Development Important? . . . . .	1
1.2 Traditional Solutions . . . . .	2
1.3 LEAP: Giving developers more control and insight . . . . .	3
1.4 Thesis Statement . . . . .	6
1.5 Evaluation of the Leap toolkit . . . . .	6
1.6 Anticipated Contributions . . . . .	8
1.7 Organization of the Proposal . . . . .	8
2 Related Work . . . . .	9
2.1 Personal Software Process . . . . .	9
2.1.1 Goals . . . . .	9
2.1.2 Learning the PSP . . . . .	10
2.1.2.1 PSP0: The Baseline Process . . . . .	10
2.1.2.2 PSP1: The Personal Planning Process . . . . .	11
2.1.2.3 PSP2: Personal Quality Management . . . . .	12
2.1.2.4 PSP3: Cyclic Personal Process . . . . .	12
2.1.3 Using the PSP . . . . .	13
2.1.4 Evaluations of the PSP . . . . .	13
2.1.5 Disney Thesis on Data Quality in the PSP . . . . .	14
2.2 Automated PSP Tools . . . . .	15
2.2.1 Full automation . . . . .	16
2.2.1.1 psptool . . . . .	16
2.2.1.2 PSP Studio . . . . .	16
2.2.1.3 PSP Tool . . . . .	16
2.2.2 Partial PSP automation . . . . .	16
2.2.2.1 pplog-mode, PPLog Control, Timmie and makelog . . . . .	16
2.2.2.2 titrax . . . . .	17
2.2.2.3 <i>timelog</i> . . . . .	17
2.2.2.4 PC LOC Accounting Tools . . . . .	17
2.2.2.5 locdelta . . . . .	17
2.2.2.6 LOCC . . . . .	17

2.3	Formal Technical Review . . . . .	18
2.4	Measurement Dysfunction . . . . .	21
2.4.1	Measurement Dysfunction in the PSP . . . . .	22
2.4.2	Measurement Dysfunction in Review . . . . .	22
2.4.3	Measurement Dysfunction in the Leap toolkit . . . . .	22
3	Supporting Software Developer Improvement with LEAP . . . . .	23
3.1	Background . . . . .	23
3.2	Design criteria . . . . .	24
3.2.1	Criteria #1: Light-Weight . . . . .	24
3.2.2	Criteria #2: Empirical . . . . .	25
3.2.3	Criteria #3: Anti-measurement Dysfunction . . . . .	25
3.2.4	Criteria #4: Portable . . . . .	25
3.3	Leap toolkit: a reference implementation of the LEAP philosophy . . . . .	25
3.3.1	Support for personal process improvement . . . . .	25
3.3.2	Support for Review . . . . .	26
3.3.3	Reducing Measurement Dysfunction . . . . .	26
3.3.4	Providing Light Weight Support . . . . .	27
3.3.5	Supporting Empirical Data Analysis . . . . .	27
3.3.6	Reducing Measurement Dysfunction . . . . .	27
3.3.7	Providing a Portable Tool . . . . .	28
3.4	Intended Benefits of Leap toolkit's design . . . . .	28
3.4.1	The Leap toolkit prevents many important classes of errors found in the PSP . . . . .	28
3.4.2	The Leap toolkit improves estimation and planning . . . . .	29
3.4.3	The Leap toolkit reduces collection stage errors . . . . .	29
4	Evaluation . . . . .	30
4.1	Claim #1: Preventing important classes of errors . . . . .	30
4.2	Claim #2: Sophisticated approaches to data analysis . . . . .	31
4.2.1	Experimental environment . . . . .	31
4.2.2	Experimental variables . . . . .	31
4.2.2.1	Independent and Dependent variables . . . . .	31
4.2.2.2	Blocking variable . . . . .	32
4.2.3	The Design . . . . .	32
4.2.4	Analysis . . . . .	33
4.2.5	Example Data . . . . .	34
4.3	Claim #3: Reduces collection stage errors . . . . .	36
4.3.1	Case Study Method . . . . .	36
4.3.1.1	Ensuring Anonymity . . . . .	36
4.3.2	Data Collection . . . . .	36
4.3.2.1	Student's raw data . . . . .	37
4.3.2.1.1	Indirect Collection Error Evidence . . . . .	37
4.3.2.1.2	Direct Collection Error Evidence . . . . .	37
4.3.2.2	Surveys . . . . .	37
4.3.2.3	Leap Survey #1 . . . . .	38
4.3.2.3.1	Topics . . . . .	38
4.3.2.3.2	Summary of Questions for Survey #1 . . . . .	38

4.3.2.4	Leap Survey #2 . . . . .	38
4.3.2.4.1	Topics . . . . .	38
4.3.2.4.2	Summary of Questions for Survey #2 . . . . .	39
4.3.2.5	Leap Survey #3 . . . . .	39
4.3.2.5.1	Topics . . . . .	39
4.3.2.5.2	Summary of Questions for Survey #3 . . . . .	39
4.3.2.6	Leap Survey #4 . . . . .	39
4.3.2.6.1	Topics . . . . .	39
4.3.2.6.2	Summary of Questions for Survey #4 . . . . .	39
4.3.3	Possible Interviews with students . . . . .	40
5	Time line . . . . .	41
A	Leap Evaluation Surveys . . . . .	42
	Bibliography . . . . .	53

# List of Tables

4.1	Example Project Data. . . . .	34
4.2	Example Predicted Error. . . . .	35
4.3	Example ANOVA for Predicted Error. . . . .	35
4.4	Proposed Study Schedule. . . . .	38
5.1	Proposed Research Time line. . . . .	41

# List of Figures

1.1	Eras of Software Development Improvement . . . . .	2
1.2	Leap Toolkit Controller. This is the main controller for the Leap toolkit. The developer may start data recording tools or start tools to modify their definitions. . . . .	5
1.3	Hee Project Viewer. This tool allows the developer to define, plan and analyze a single project. Cam has filled out the name, description and start date for the Multi-User Calendar project. He has also decided to use his Development process for this project. . . . .	5
1.4	Io time recording tool. Io allows the developer to easily record the amount of time they spend working on a task. They may also account for any interruptions by recording interrupt time. In this Figure Cam has worked for 40 minutes on the design of the Multi-User Calendar. . . . .	7
1.5	Time Estimation Tool. The time estimation tool shows Cam's historical data. Cam has chosen Linear Regression for the trend lines and Lines of code as the size measures for time estimation. The planned size 3022 is taken from Hee. Based upon this data the project should take from 1699 to 2321 minutes. . . . .	7
2.1	PSP levels . . . . .	11
2.2	PSP0.1 phases . . . . .	12
2.3	PSP2.0 phases . . . . .	12
2.4	PSP3.0 phases . . . . .	13
2.5	Generic Review Process . . . . .	18
2.6	Spectrum of Formal Technical Reviews . . . . .	20
A.1	Leap survey #1 page 1 . . . . .	43
A.2	Leap survey #1 page 2 . . . . .	44
A.3	Leap survey #2 page 1 . . . . .	45
A.4	Leap survey #2 page 2 . . . . .	46
A.5	Leap survey #2 page 3 . . . . .	47
A.6	Leap survey #3 page 1 . . . . .	48
A.7	Leap survey #3 page 2 . . . . .	49
A.8	Leap survey #4 page 1 . . . . .	50
A.9	Leap survey #4 page 2 . . . . .	51
A.10	Leap survey #4 page 3 . . . . .	52

# Chapter 1

## Introduction

*At the start, when we know much about the problem and nothing about the solution, the solution is very abstract.* – Robert H. Dunn

Every software developer wishes they got home from work sooner and spent less of their weekend at work. Software developers work very hard and very long, yet software is often delivered late, over budget, and full of defects. Over forty years of software development experience has not helped us solve this problem. How can software developers gain more control over their software development and produce high quality software efficiently? Project LEAP and the Leap toolkit in particular attempts to give developers the control and skills they need. Before I discuss Leap I will give a short background of the software development problem.

### 1.1 Why is Quality Software Development Important?

Software is controlling more safety critical tasks and important functions[1]. Yet software errors occur sometimes with horrible costs. Between 1985 and 1987 the Therac-25 radiation therapy machine killed two people and seriously injured four others by delivering massive radiation overdoses. Investigations found that many issues were to blame including faulty software[24].

In March 1995, the Denver International Airport opened over 16 months late and over 100 million dollars over budget. One of the primary reasons for the delay and overrun was the presence of major bugs in the baggage handling control software[11].

Another problem with current software development is productivity. The insatiable demand for more software has out-paced our ability to produce software. Software productivity has not kept pace with hardware cost/performance ratios. The most optimistic rate at which programmer productivity is increasing is 5% per year, while there is greater than a ten fold increase in the



demand for software each decade.[6] The government of the United States of America faces this problem. In 1996 Computerworld reported that delays in overhauling the federal tax computer systems cost the U.S. Treasury as much as \$50 billion per year.[11] The problem of overhauling of the tax computers is not purely a software issue but the software system is a large part of the problem.

## 1.2 Traditional Solutions

Software developers and managers have addressed software quality and development issues since the beginning of the computer age. Developer and managers have continuously augmented the development methods they use. We can divide these methods into four eras: hope-based, product-based, organization-based, and individual-based. Figure 1.1 shows a time line of software development improvement and the different eras.

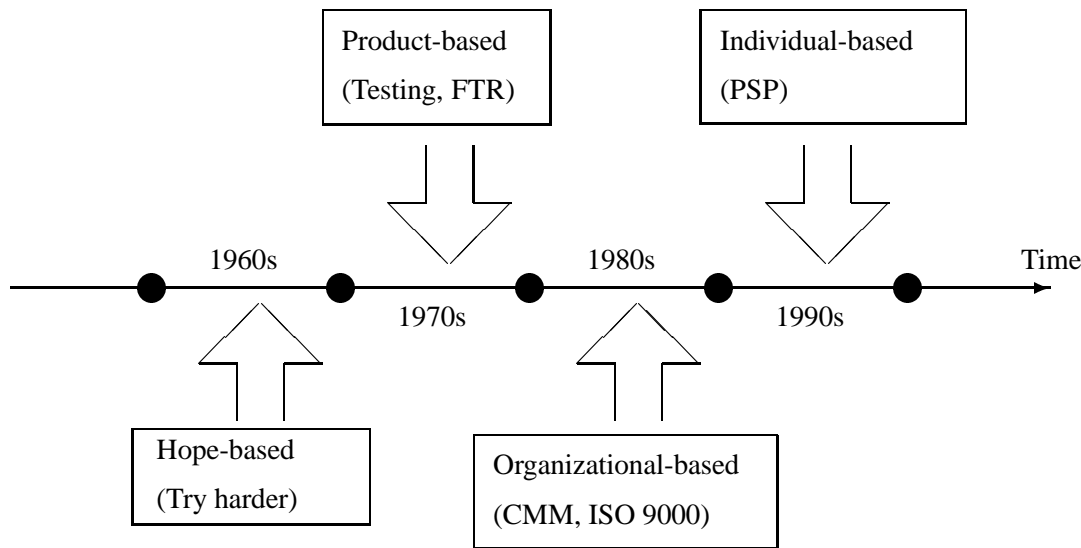


Figure 1.1: Eras of Software Development Improvement

In the 1960's much of the software development improvement efforts were just focused on doing better. The problems of software development tended to be small since computers were very small as compared to today's computers. Trying harder seemed like a reasonable solution.

In the 1970's people realized while the "try-harder" method did help improve software development it was not enough. Computers and software programs were getting more complex and

new methods were needed. Developers and researchers started looking at the work products. Many Developers and Computer Scientists advocated testing to help improve the quality of software. In 1976 Fagan reported on Software Inspection's[8] success as a method to efficiently improve the quality of the work product.

In the late 1980's the focus shifted from the work products to the organizations that produced the work products. The SEI introduced the Capability Maturity Model[26] and ISO 9000[17] became wide spread. These organizational processes helped improve software quality and the development processes but didn't completely solve the problem.

In the late 1990's some Computer Scientists and Developers changed their focus again from the organization to the individual software developer. In 1995 Humphrey introduced the Personal Software Process[15], a software development process and improvement process for individual software developers. The PSP is a manual process where the developer collects data about the amount of time they spend, the size of the work product and the defects they make while developing software. By analyzing the data at the end of the project, the developer gains insights into their development process. These insights help the developer increase productivity and work product quality.

Many studies have shown the PSP helps improve software development[9, 13, 23, 30], but is not the complete solution to the software development issue.

After using the PSP for two years in our research group, the Collaborative Software Development Laboratory (CSDL), we decided to try to build upon the PSP's strong foundation and incorporate features of formal technical review to produce a more effective software developer improvement tool.

### **1.3 LEAP: Giving developers more control and insight**

LEAP is a design philosophy intended to produce effective tools that allow developers to gain valuable insight into their own software development. Using the LEAP design philosophy I developed the Leap toolkit, a Java application that supports technical skill acquisition.

#### **Using the Leap toolkit example**

I will introduce the Leap toolkit by using a hypothetical software developer, Cam, and his manager, Philip. Cam and Philip work for a small world class Java software development company.

Cam has been using the Leap toolkit for about a year to keep track of his Java programming projects. He has a small database of over 30 Java projects.

### **Starting a new project**

Philip calls Cam into his office to discuss Cam's next project. The next project is an extension to their single user calendar tool that allows multiple users to use the same calendar while keeping some events private. Philip gives Cam the requirements for the new extension and ask him how long the project will take. Cam tells Philip that he needs to do some design work before he can give an accurate estimate. Cam goes back to his cubicle and starts the Leap toolkit shown in Figure 1.2.

### **Defining the new project**

To start the new project Cam opens the Projects (Ilio) tool. To create a new project Cam starts the project editing tool Hee on the first blank line. He types in the name of the new project "Multi-User Calendar" and a brief description of the project. Then he selects the start date for the project and chooses the PhaseSet that he plans on using. The PhaseSet is a set of phases that describe his development process. Cam evolved his current PhaseSet after experimenting with different development processes. Figure 1.3 shows the Hee project viewer.

### **Developing an initial design**

Cam then starts the Io timer tool to record the time he spends designing the new project. Figure 1.4 shows the Io timer after he started recording time for the design phase.

Cam works on the design for the new project and when he finishes his design he clicks the stop button on Io and then records his time in the Leap toolkit.

### **Project Planning**

With the initial design of 30 classes and 168 methods.

**Size Planning** Cam then opens the Project Comparisons tool to find out his average lines of code per method. His average is 17.99 lines of code per method so he calculates that the whole project will be 3022 lines of code. He reopens the Hee project viewer for the Multi-User calendar project and enters in his planned sizes.



Figure 1.2: Leap Toolkit Controller. This is the main controller for the Leap toolkit. The developer may start data recording tools or start tools to modify their definitions.

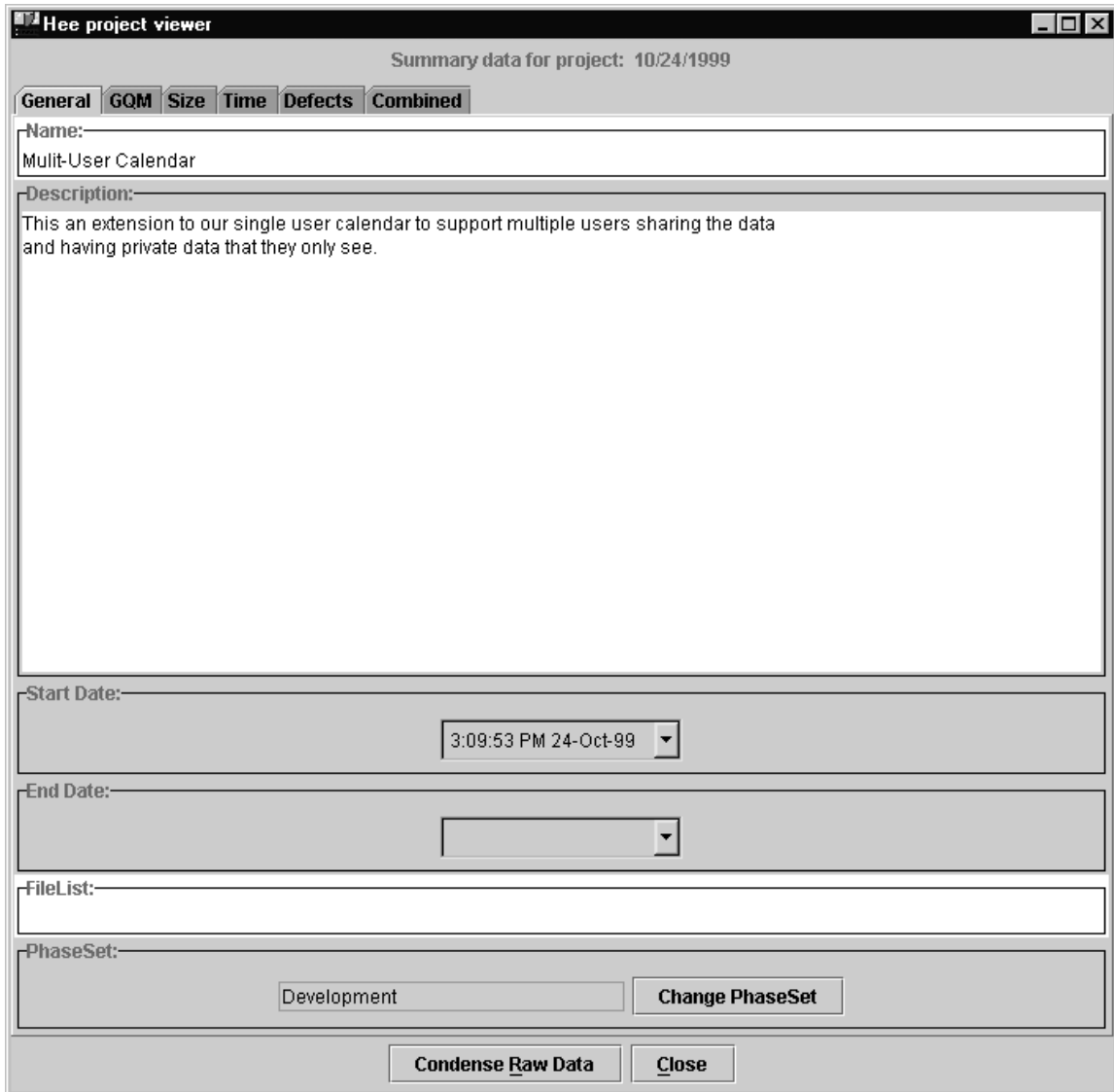


Figure 1.3: Hee Project Viewer. This tool allows the developer to define, plan and analyze a single project. Cam has filled out the name, description and start date for the Multi-User Calendar project. He has also decided to use his Development process for this project.

**Estimating effort** He then goes to the time tab in Hee and starts the time estimation tool. Cam choose to use his historical average rates for the trend lines and lines of code for the size grain size then presses the estimate button. Figure 1.5 shows the time estimation tool with the estimate. Based upon his historical data this project will take about 2010 minutes. Cam tries some of the other combinations like methods and linear regression model to give him a range of estimates. Based upon all these estimates, Cam estimates that it will take him anywhere from 28 1/3 hours to 38 2/3 hours of direct work to complete the project. Cam enters in his time estimate into Hee and then sets up a meeting with Philip.

### **Negotiation**

At the meeting with Philip, Cam tells Philip that the project will take him between two and three weeks. From Cam's historical data he knows that he only gets in an average of 3 direct hours per project per work day so the 28 1/3 direct hours will take 9 1/2 days to complete. Philip wants the project done in one week. Cam says that this is only possible if he can stop working on his other projects and focus solely on the Multi-User Calendar project. Philip agrees that Cam may drop his other projects until the calendar is finished.

## **1.4 Thesis Statement**

LEAP provides a more accurate and effective way for developers to collect and analyze their software engineering data than methods designed for manual enactment.

## **1.5 Evaluation of the Leap toolkit**

I will evaluate the main thesis of this work by breaking the thesis down into three claims.

First, the Leap toolkit prevents many important errors. To evaluate this claim I will discuss the design and automation features in the Leap toolkit that prevent these errors from occurring.

Second, the Leap toolkit provides data analysis that is not practical with a manual method. To evaluate this claim I will conduct an empirical experiment to determine if any particular time estimation technique is more accurate. Using the Leap toolkit I will compare 13 different time estimation techniques and determine if any of them are more accurate in predicting the effort for a new project. In a manual method, such as the PSP, only one time estimation method can be provided.



Figure 1.4: Io time recording tool. Io allows the developer to easily record the amount of time they spend working on a task. They may also account for any interruptions by recording interrupt time. In this Figure Cam has worked for 40 minutes on the design of the Multi-User Calendar.

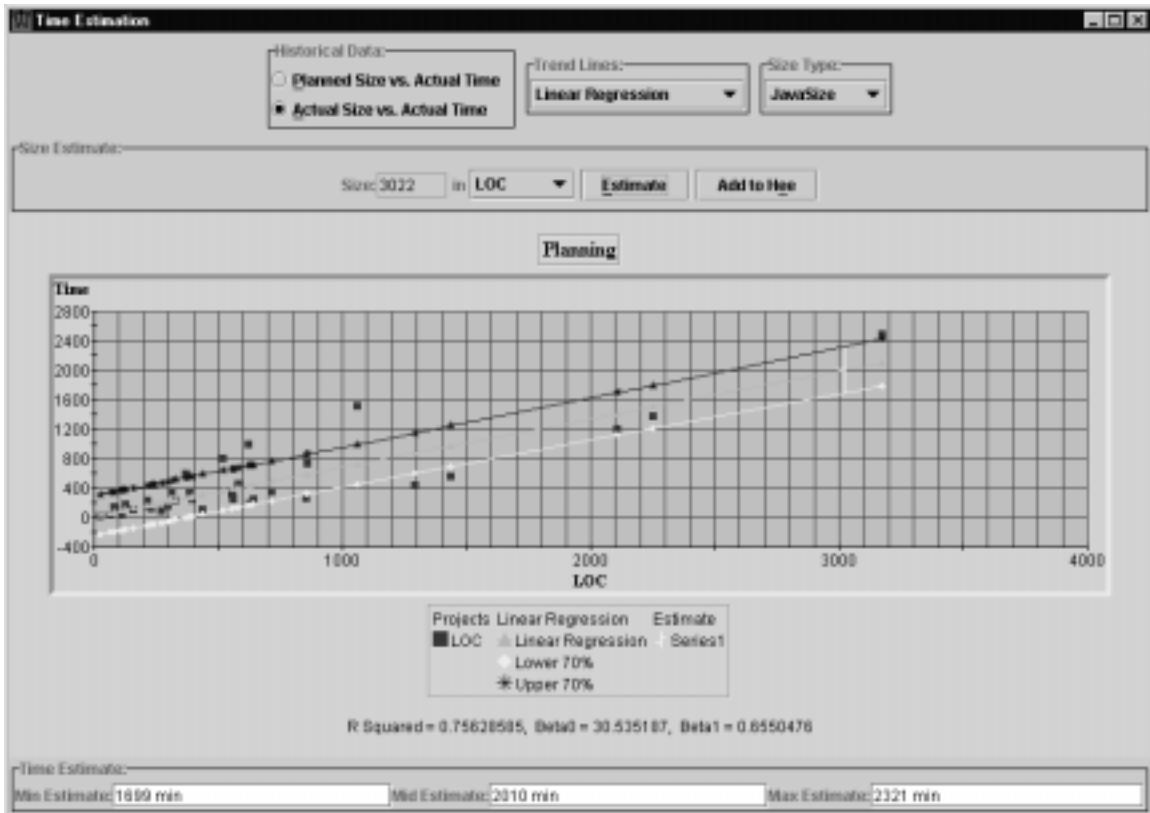


Figure 1.5: Time Estimation Tool. The time estimation tool shows Cam's historical data. Cam has chosen Linear Regression for the trend lines and Lines of code as the size measures for time estimation. The planned size 3022 is taken from Hee. Based upon this data the project should take from 1699 to 2321 minutes.

Third, The Leap toolkit reduces the level of collection stage errors. To evaluate this claim I will conduct four surveys and analyze the software development data collected by graduate students in an advanced software engineering course at the University of Hawaii.

## **1.6 Anticipated Contributions**

This research is designed to produce several valuable contributions to the software engineering community. One major contribution is the Leap toolkit. I have made the Leap toolkit freely available on the Internet. Software developers may down-load the Leap toolkit and use it in their own work. The Leap toolkit has been available for over one year and many developers have down-loaded it. The Leap toolkit also provides a novel tool for software developer education, as is being demonstrated in a graduate level class in software engineering this semester. Instructors can gain insight into how the students are actually spending their time and provide more detailed help.

Another anticipated contribution is the results of the time estimation experiment. The results may indicate that one estimation technique is more accurate, or that different techniques are more accurate for different people, or that no technique is significantly better than any other. If it turns out that there is no more accurate estimation technique then developers can use simple averages which are easy to calculate instead of complex formulas. If there is a best estimation technique then developers can adopt it and gain more accurate time estimations.

## **1.7 Organization of the Proposal**

This proposal is organized as following: Chapter 2 relates the current research to the broader context of existing work. Chapter 3 depicts the main design features and planned benefits of LEAP. Chapter 4 outlines the evaluation methods I plan to use to evaluate the effectiveness of LEAP. Finally, Chapter 5 presents the current research plan.

# Chapter 2

## Related Work

*If you don't know what you are doing, it is hard to improve it.* – Watts Humphrey

LEAP is a result of our experience using the Personal Software Process (PSP) for over three years, our experience with Formal Technical Review (FTR) and our attempts to improve the quality of software development. This chapter briefly discusses the PSP, some of the different tools developed to support the PSP, and the software quality assurance process called Formal Technical Review. The chapter concludes with a discussion of measurement dysfunction and some of the data quality issues found in the PSP and FTR.

### 2.1 Personal Software Process

The Personal Software Process[15] is a self-improvement process for software developers. In his book “A Discipline for Software Engineering” Watts Humphrey teaches software developers how to become their own software development coaches. Sports coaches observe the performance of their players, evaluate their performance, then make suggestions for improvement. This is the classic *observe, evaluate, modify* cycle for improvement. Software developers using the PSP can become their own coaches. The developer records how they develop software and after each project they self-evaluate how they performed. These evaluations should lead to improvements on future projects. In other words, The software developer conducts a longitudinal case study of their own development process. They can initiate changes and observe the effects of those changes.

#### 2.1.1 Goals

Two main goals of the PSP are



- to produce high-quality software as efficiently as possible and
- to improve the developers ability to estimate the amount of effort required to produce the software.

These two goals drive the whole PSP. The data collection and analyses are focused on improving the developer's software development and estimation skills.

## **2.1.2 Learning the PSP**

To teach developers how to use the PSP, Humphrey defines seven PSP processes (0, 0.1, 1.0, 1.1, 2.0, 2.1, 3.0). Each process has detailed scripts telling the user exactly how to perform the process. Figure 2.1 shows the seven levels. Exercises at the end of each chapter in "A Discipline for Software Engineering" ask the reader to use the knowledge from the chapter to improve their development skills. The chapters introduce powerful development techniques: design and code reviews, size and time estimation methods, and design templates. These techniques help the developer produce high quality products efficiently. As developers go through the book they develop 10 small software projects using the different PSP levels.

### **2.1.2.1 PSP0: The Baseline Process**

The baseline processes PSP0 and PSP0.1 introduce the concepts of data collection and size measurement to the developer. The purpose of these processes is to give the developer a basis for their improvement. The developer learns exactly how they develop software. They learn to use the Time Recording Log, Defect Recording Log and Postmortem forms to record and analyze time, size and defect data. In these two processes the developer uses the planning, design, code, compile, test and postmortem phases. Figure 2.2 shows the order of the phases.

In the planning stage they make their "best guess" as to how long the project will take. In the postmortem phase they fill out the Project Summary form. PSP0 uses four scripts, six phases and three forms.

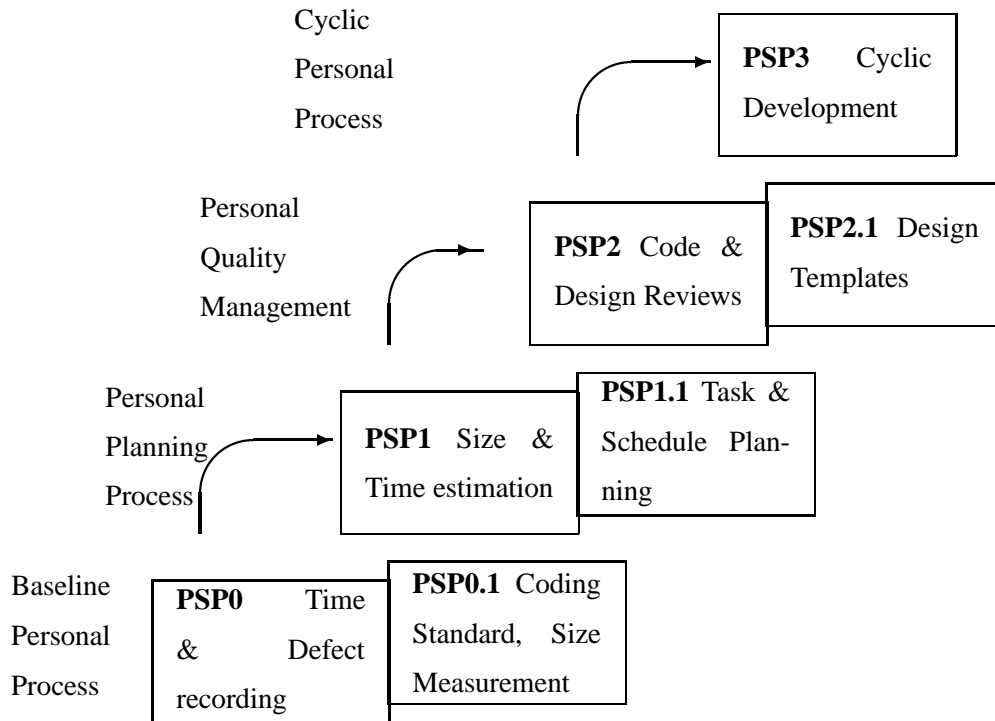


Figure 2.1: PSP levels

### 2.1.2.2 PSP1: The Personal Planning Process

In PSP1 the developer adds a detailed Planning phase to their development. In the planning phase they make explicit, documented plans for their work. During the postmortem phase they compare their plan to their actual performance. PSP 1.1 adds the concepts of task and schedule planning. This allows the developer to better estimate and schedule their projects. One of the purposes of these PSP levels is to show that developers can control and predict their development process. PSP1 uses four scripts, six phases and six forms.

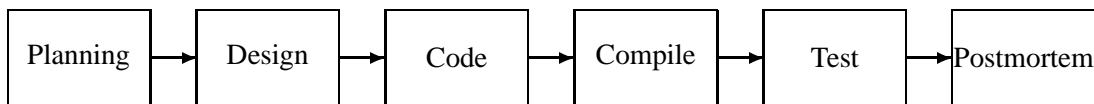


Figure 2.2: PSP0.1 phases

### 2.1.2.3 PSP2: Personal Quality Management

PSP2 introduces quality control measures by adding two reviews to the process that the developer uses. The developer reviews their own design before they start coding and they review their code before they start compiling. These reviews should catch defects earlier and reduce the cost of fixing defects. PSP2.1 addresses the design process by introducing design templates, logic and state diagrams these tools should help the developer produce more correct programs with less overall effort. The purpose of PSP2 is to provide the developer with tools to efficiently improve the quality of their work products. PSP2 uses four scripts, eight phases and twelve forms.

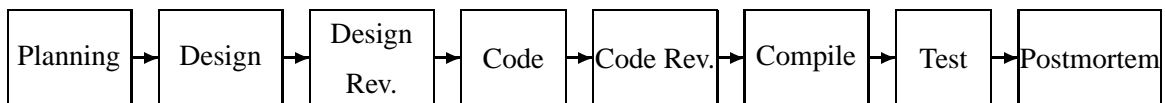


Figure 2.3: PSP2.0 phases

### 2.1.2.4 PSP3: Cyclic Personal Process

PSP3 changes the overall development process from a strict linear waterfall model to a cyclic spiral model. PSP3 allows the developer to subdivide a larger program into smaller pieces that are developed using PSP2. Figure 2.4 shows the new development process. The whole program is built up of enhancements on the previously completed increments. This builds up a high quality final product as long as each increment is of high quality. The purpose of PSP3 is to expand the PSP to larger projects. PSP3 uses six scripts, ten phases and twenty forms.

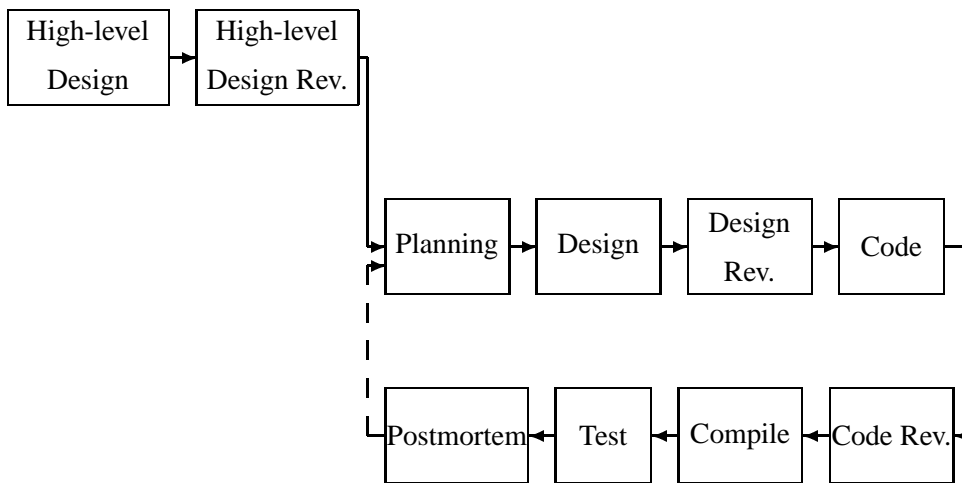


Figure 2.4: PSP3.0 phases

### 2.1.3 Using the PSP

There is a distinction between the PSP and the way the PSP is taught. Humphrey says that the PSP should be modified by the user to support their own goals and situation. However, the developer should not modify the process of learning the PSP — they must go through all the stages to learn how to properly use the PSP before they modify the process.

Instead of trying to modify the PSP, most PSP users just choose one of the PSP levels. One reason that it is so difficult to modify the PSP is that the two goals of the PSP are so intertwined into the forms and scripts of the PSP that changing the goals would require a major overhaul of the forms and scripts.

One of the motivations behind Leap is the user should be able to easily modify their process and not be forced to use a process they do not like. If the user wants to use PSP3, they may. If they want to drop the high-level design review phase, then they may without requiring a dramatic redesign of the method.

### 2.1.4 Evaluations of the PSP

In a 1996 article, Watts Humphrey reported the results of 104 engineers taking the PSP course[16]. He states that the two goals of PSP were met. First, reported defects fell from an average of 116.4 defects per thousand lines of code (KLOC) for assignment 1 to 48.9 defects per KLOC for

assignment 10. Second, the estimation accuracy of the students increased. For assignment 1 32.7% of the engineers' estimates were within 20% of their actual times. By assignment 10 49.0% of the engineer's estimates were within 20actual times.

In 1996, Sherdil and Madhavji studied human-oriented improvement in the Software Process[31]. They used PSP as a basis for their studies. They found that subjects reduced their defect by 13% after project 6, when code reviews are introduced. They also found that their subjects reduced their size estimation error by more than 7% than expected.

Hayes and Over conducted an extensive study, with 298 engineers, of the PSP[13]. The results of the study were impressive. Over the projects completed, the median improvement in size estimation was a factor of 2.5. This means that 50% of the engineers reduced their size estimation error by a factor of 2.5. The median improvement in time estimation was 1.75. The median reduction in overall defect density was by a factor of 1.5. The engineers substantially reduced the percentage of defects surviving to later stages of development.

Pat Ferguson and others report excellent results with PSP adoption at Advanced Information Services, Motorola and Union Switch and Signal[9]. However, Barry Shostak and others report poor adoption of PSP in industry[32, 7].

Andrew Worsley reports on his own impressions of the PSP after completing all 10 assignments[36]. He found an improvement in his defect density, but at the cost of productivity.

All of the above studies assumed that the data recorded by the subjects using the PSP was accurate and correct. Anne Disney conducted a study to see if this assumption was correct.

### **2.1.5 Disney Thesis on Data Quality in the PSP**

Anne Disney did her masters thesis on data quality issues in the PSP. She found that in her sample of students who learned the PSP, the errors in their data were significant. These errors lead to incorrect insights into the students development practices. For example, in several cases the students' incorrect data indicated that they were over estimating their yield when in fact they were underestimating their yield.

In her thesis, Disney classified the data errors found in the PSP data into seven categories:

- **Calculation Error:** This error type applied to data fields whose values were derived using any sort of calculation and the calculation is done incorrectly. In her study 46% of all the errors were calculation errors.

- **Blank Fields:** This error type applies to data fields that are required but not filled in. 18% of all the errors in the study were blank fields.
- **Inter-Project Transfer Error:** This error type applies to data fields whose values involved data from a prior project and the value is not the same as the prior project's value. Inter-project transfer errors accounted for 14% of the errors in Disney's study.
- **Entry Error:** This error type applies to fields where the user clearly does not understand the purpose of the field or used an incorrect method in selecting data. 9% of the errors were entry errors.
- **Intra-Project Transfer Error:** This error type applies to data fields whose values involve other data fields in the same project, but are incorrectly filled in. 6% of the errors were intra-project transfer errors.
- **Impossible Values:** This error type indicates that two values were mutually exclusive. 6% of the errors were impossible values.
- **Sequence Error:** This error type is used to indicate when the user moved back and forth between phases. Only 1% of the errors were sequence errors.

She found that 34% of the errors made affected multiple forms and multiple projects. This means that an error in an earlier project rippled through the future projects affecting the student's PSP data. One possible solution is automated tool support to reduce data errors. Human beings will make mistakes in any process. Automating much of the data entry and transfer will reduce the opportunity to make mistakes.

## 2.2 Automated PSP Tools

Soon after the PSP was introduced many developers answered the challenge of automating the PSP. Some developers automated the entire PSP while others just automated different aspects of the PSP.

## 2.2.1 Full automation

### 2.2.1.1 psptool

psptool by Andrew M. Worsley[29] is a tool that runs under X/Unix or on Win32S platforms. It allows the user to collect size, time and defect data. It also produces a PSP2.1 like plan summary and supports time estimation based upon historical data and an initial size estimate.

### 2.2.1.2 PSP Studio

PSP Studio from Eastern Tennessee State University's Design Studio 1997[14] automates all the PSP levels. It runs on Win32 platforms and supports all the PSP levels from 0 through 3.0. It produces all the postmortem reports after the projects are complete.

### 2.2.1.3 PSP Tool

PSP Tool[4] from Anne Disney, is written in Progress 4GL/RDBMS and runs on SCO Unix. It implements the PSP0, PSP0.1, and PSP1 completely while the higher levels are not fully implemented. The PSP Tool allows the user to define their own *Defect models*. *Defect models* refer to specific defects with in a defect type. The user may enter a defect model in the defect recording tool and it will fill in the fields. This reduces the mental overhead of the user and speeds up defect recording.

## 2.2.2 Partial PSP automation

Developers using the PSP record three types of primary data: time, size and defects. Many tool developers have developed tools to automate the collection of one or more of these primary metrics. The following tools focus on collecting the raw data and not enforcing the entire PSP process.

### 2.2.2.1 pplog-mode, PPLog Control, Timmie and makelog

Researchers at the University of Karlsruhe developed several tools that help automate the collection of time and defect information[28]. Their data collection tool, pplog-mode.el is an extension for GNU Emacs[12]/XEmacs[37], powerful text editors used by programmers. The developer using these tools can record their time and defects that they find while using Emacs. Users defines a *logging key*. When the *logging key* is pressed Emacs automatically switches to the *logging buffer*

where the user may type in a description of the event that just occurred. Emacs automatically inserts the time-stamp of the event. The data is saved in a database file. To analyze the data files the researchers wrote a PERL script called evalpsp.

The researchers wrote additional tools for recording data in formats that evalpsp could analyze. They are PPLog Control, a full-featured GUI application for win32 machines, Timmie, a multi-day, multi-project Java application for recording time and defect data, makelog, a command line program for PC users similar to pplog-mode.el.

#### **2.2.2.2 titrax**

titrax[34] is a time tracker by Harald T. Alvestrand. It is written in C and runs under X. It allows the user to record their times and includes some simple time analysis tools.

#### **2.2.2.3 timelog**

*timelog*[33] by Christoph Clemens Lahme is a Java program that allows the user to record time.

#### **2.2.2.4 PC LOC Accounting Tools**

PC LOC Accounting Tools[28] by Christian Segor are three tools one that inserts tags into source code, one to count the lines of code LOC, and one to remove the tags from the source code. The counter is able to count base LOC, modified LOC, added LOC and deleted LOC.

#### **2.2.2.5 locdelta**

locdelta[28] is a perl script that calls a user supplied program to format the source code then calls the Unix diff program to count the base, modifies, added and deleted LOC.

#### **2.2.2.6 LOCC**

LOCC[25] written by Joe Dane is an extensible system for producing hierarchical, incremental measurements of work product size. LOCC can produce output files that the Leap toolkit can use.

The Leap toolkit builds upon the ideas of the PSP. It can record all of the data needed in the PSP and yet, it does not require the user to record all three types of data for interesting analyses. The Leap toolkit is more flexible than the fully automated PSP support tools, but it does not enforce



the PSP processes like they do. The Leap toolkit currently does not support all of the reports that the PSP processes require. I can easily add these reports to the Leap toolkit if users want them. The next section discusses the second source of inspiration for LEAP, Formal Technical Review.

## 2.3 Formal Technical Review

Formal Technical Review is defined as

a method involving a structured encounter in which a group of technical personnel analyzes an artifact according to a well-defined process. The outcome is a structured artifact that assess or improves the quality of the artifact as well as the quality of the method.[10]

The technical personnel that analyze the work product may fulfill many different roles. The generic roles in any FTR are author, moderator, reviewer, scribe, and leader. The author is the person who created the artifact under review. The moderator moderates the group meetings that may be held during the review process. The reviewers are the technical people who analyze the work product. The scribe records all the issues found by the reviewers. The leader organizes the entire process. During any review an individual may perform many of these roles.

All formal technical reviews follow the same generic process. Figure 2.5 shows the generic FTR process. Many organizations modify the generic process, but it is the basis for FTR.

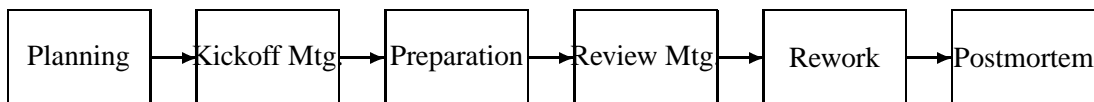


Figure 2.5: Generic Review Process

In the planning phase the review leader plans the review. They gather the review materials: work product, guidelines, checklists, standards, etc. They choose the review members and schedule the meetings and deadlines. Another important part of the planning phase is to determine the goals of the review. The goals of the review help determine the level of formality and the process to use.

Once the planning is done a Kickoff meeting is often held to orient all the review members to the goals of the review and distribute the review materials. The Kickoff meeting is often not used if the review members are familiar with the work product and review process.

In the preparation phase the reviewers familiarize themselves with the work product. In some review methods like Inspection[8], the reviewers do not record any issues, but just become familiar with the work product. In other methods like FTArm[21, 22, 18, 19, 20, 35] the reviewers record their issues.

In the Review Meeting phase the reviewers gather to discuss the work product. Often the moderator proceeds through the work product and the reviewers raise any issues they have with the work product. The output of the Review Meeting phase is a consolidated list of all the issues found by the reviewers.

During the Rework phase the author of the work product takes the consolidated list of issues and addresses each issue. Some issues may require rework, others may not be defects. The author fixes all the defect that they can.

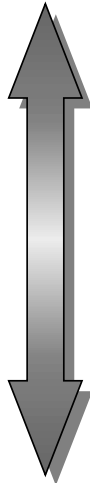
In the last phase, Postmortem, the review team evaluates the entire review process including the reworked work product. Often the review team approves the work product or decides that it should be re-reviewed. The review process is also analyzed to generate suggestions for improvement.

This generic review process covers a wide spectrum of different review styles. Figure 2.6 summarizes the range of the different review methods.

The most informal reviews are called Walkthroughs[38]. In walkthroughs there is very little preparation. The members in the walkthrough gather together and the author walk the group through the artifact explaining what the artifact does. As the author walks through the artifact the reviewers are looking for problems in the artifact. When a problem is discovered, it is often fixed right there in the walkthrough. Walkthroughs often combine the Kickoff meeting, Review meeting, Rework phase, and Postmortem phase all into one meeting. The author is often the moderator, leader and scribe.

More formal reviews are known as Technical Reviews. Technical Reviews are more formal and normally follow all six phases of the review process. The goals of technical reviews may not focus purely on evaluating the work product and can include team building and developer education.

The most formal reviews are Inspections[8]. Inspections have one primary goal, to detect and remove defects from the work product effectively and efficiently. To accomplish this goal the process is very formal and discussion during the Review Meeting is solely focused on reporting de-



<b>Method Family</b>	<b>Typical Goals</b>	<b>Typical Attributes</b>
<b>Walkthroughs</b>	Developer training Quick turnaround	Little/no preparation Informal process No measurement
<b>Technical Reviews</b>	Requirements elicitation Ambiguity resolution Team building	Formal process Author presentation Wide range of discussion
<b>Inspections</b>	Detect and remove defects efficiently and effectively	Formal process Checklists Measurements Verify phase

Figure 2.6: Spectrum of Formal Technical Reviews

fects not solutions to defects. The moderator must control the meeting to ensure that the discussion does not wander.

In the PSP Watts Humphrey introduced the concept of a single person review. The developer conducts a technical review of their work product to detect defects and improve the quality of the work product. The PSP reviews are formal and similar to Inspections since the developer uses checklists to help guide their focus. Combining the PSP's personal reviews and group reviews should help improve the work product and help improve the developer. The defects that the group detects can be analyzed to provide additional insights for the developer.

The Leap toolkit supports group review of work products by allowing reviewers to send the defects they find to each other over the Internet. The author of the work product can collect all the reviewers' defects to fix them and also combine those defects with the defects the author found during development. By combining the ideas of PSP and FTR the Leap toolkit provides the developer with more insight. The reviewers will find defects that the developer misses. The developer can use this data to learn more about their development process.

Whenever data is collected about a process, as in the PSP and FTR, the question of what do you do with this data arises. The use of measurement data raises the issue of Measurement Dysfunction.

## 2.4 Measurement Dysfunction

Robert Austin introduces the term “Measurement Dysfunction” in his book “Measuring and Managing Performance in Organizations”[2]. He defines dysfunction as “the actions leading to it fulfill the letter but not the spirit of stated intentions.” In measurement dysfunction, people try, consciously or unconsciously, to change a measure used for evaluation, without trying to change the actual underlying behavior or result that is being measured. The fundamental problem with measurement is that it is impossible to fully measure a behavior or activity. So when people focus on the letter of the measurement they may ignore an important part of the behavior, thus reducing their overall effectiveness.

Austin cites an apocryphal example of measurement dysfunction, a Soviet boot factory. The boot factory was evaluated by the number of boots produced. To meet their quota of boots the factory managers produced only left boots, size 7, since by producing only left boots in one size they could maximize the total output of the factory. Austin uses a study of an employment office by Peter Blau in 1963[3] to provide more insight into measurement dysfunction. The goal of the employment office was to find jobs for their unemployed clients. The employment office employees were evaluated primarily by the number of interviews conducted. The employees responded by focusing as much time as possible on doing interviews, and very little time in finding jobs for their clients. This behavior resulted in client receiving fewer job referrals. When the management changed the evaluation measure to include eight different indicators the employees changed their behavior to improve their standing against various indicators. Some employees destroyed records of interviews that did not result in job referrals and made referrals for clients that did not match the job. In both these situations the true performance of the organizations declined while the measured performance increased.

Austin divides measurements into two categories *motivational measurements*, which are used to affect the people who are being measured, and *informational measurements*, which are used for their logistical, status, and research information they convey. Motivational measurements may lead to measurement dysfunction since the people affected will focus on those measures and change their behavior. A problem with individual measures is they may be used for both motivation and information. Once a measure is taken and recorded managers can use it for status purposes or for evaluation.

### **2.4.1 Measurement Dysfunction in the PSP**

The data collected in the PSP provides valuable insight into the developer's development process. The developer learns their development rate, the types of defects they make most often, their average direct hours of work per day, and many other statistics that management could use to evaluate their performance. If management uses this data to evaluate their employees the employees may start to change their behavior to improve their measures. For example if management says developers should produce 50 lines of code per hour and the developer is only producing 40 lines of code per hour, they might stop optimizing their code since it takes time and reduces the number of lines of code.

### **2.4.2 Measurement Dysfunction in Review**

Measurement Dysfunction can also occur with review data is used to evaluate the reviewers. If management want more important defects to be discovered during the review they might want to raise the average severity of defects found during review. This might lead to reviewers categorizing all the defects they find as critical.

There are many different possible types of measurement dysfunction in review data. Some of the typical ones are defect severity inflation, preparation time inflation, and defect severity reduction. In defect severity reduction the work product is nearing a milestone and cannot pass the milestone with any sever defects. The reviewers feel pressure to keep the project on time so they reduce the severity of defect so that the project can stay on track. Defect become enhancements that will be corrected before the product is released.

### **2.4.3 Measurement Dysfunction in the Leap toolkit**

The Leap toolkit does not eliminate any of the above sources of measurement dysfunction. However, the design of the Leap toolkit addresses measurement dysfunction by allowing the user full control over the data collected and shared by the Leap toolkit.

The next chapter discusses how LEAP support software developer improvement by incorporating ideas from the PSP, FTR and addresses the Measurement Dysfunction issue.

# Chapter 3

## Supporting Software Developer Improvement with LEAP

*Measures of productivity do not lead to improvement in productivity.* - W. Edwards Deming

This chapter discusses Project LEAP and the Leap toolkit. It starts with a brief summary of why I started work on Project LEAP. Then it discusses the design criteria for LEAP compliant tools. It next discusses the Leap toolkit a reference implementation of the LEAP design philosophy. Finally, it introduces three intended benefits of the Leap toolkit.

### 3.1 Background

After using the PSP for over two years, I noticed three general problems with the PSP. First, I started to question the quality of the data recorded. I noticed that I did not record all of our defects, in part because the overhead of recording each defect is too expensive. Anne Disney and Philip Johnson conducted a study to look at the data quality of PSP data. They found that there are significant data quality issues with manual PSP.[4, 5]

Second, my experiences with industrial partners, management practices and Robert Austin's book "Measuring and Managing Performance in Organizations"[2] made me think about the issues of measurement dysfunction in PSP and review data. An organization may pressure their members to produce "good" results. There are many ways that the members can manipulate the personal data collected in the PSP to get the "right" results.

Third, after four years, the results with adoption of PSP are mixed. Pat Ferguson and others report excellent results with PSP adoption at Advanced Information Services, Motorola and

Union Switch and Signal[9]. However, Barry Shostak and others report poor adoption of PSP in industry[32, 7]. No research has been published that studies the “long term” adoption of the PSP — i.e., whether or not users trained in the PSP are continuing to use it six months, a year, or more after the training.

These issues motivated me to begin designing an automated, empirically based, personal process improvement tool. My goal is to reduce the collection and analysis overhead for the engineer, and the measurement dysfunction of the collection process. This should improve the benefits to the engineer and the long term adoption of empirically based process improvement. To pursue this work, I initiated Project LEAP, <<http://csdl.ics.hawaii.edu/Research/LEAP/LEAP.html>>, and began developing the Leap Toolkit, <<http://csdl.ics.hawaii.edu/Tools/LEAP/LEAP.html>>.

## **3.2 Design criteria**

As part of my initial research, I hypothesized that improved support for software developer improvement would be obtained by attempting to satisfy four major design criteria: light-weight, empirical, anti-measurement dysfunction, and portable.

### **3.2.1 Criteria #1: Light-Weight**

The first principle is that any tool or process used in software developer improvement should be light weight. This means that the tool or process should not impose overhead on the developer. Data collection should be easy to perform and should not add significant effort to the process. The processes that are used, should not impose a burden on the developer. We do not want the developer to worry about the improvement effort while they are doing the development. They should be worrying about the development. Analyses and other work should also require as little effort by the developer as possible. The benefit of using the improvement processes should outweigh the cost of to the developer.

This principle implies that any improvement process must be automated as much as possible. A manual process requires too much overhead by the developer. The overhead of recording information by hand and manually doing the analyses will out weigh the benefits of the process. The PSP suffers from this.

### **3.2.2 Criteria #2: Empirical**

We believe in empirical data collection the improvements should be based upon the developer's experiences. We want the developer to use the observe, evaluate, modify method for improvement. Each modification is then tested by further observation to see if the change is actually an improvement or just a false start. By using looking at their development empirically the developer is able to judge for themselves what is best.

### **3.2.3 Criteria #3: Anti-measurement Dysfunction**

Based upon my experiences as a summer intern and Richard Austin's book *Measuring and Managing Performance in Organizations*[2], I believe that any process improvement method should deal with the issue of measurement dysfunction. The empirical data collected could be misused. This issue is important since the development process is very interesting to people other than the developer. If there is measurement dysfunction then the data collected and analyses will not reflect reality. Any insights gained from this data and analyses will be faulty and may cause more problems than they solve.

### **3.2.4 Criteria #4: Portable**

Software developers often change jobs and the tool support for their development improvement should be portable. They should be able to take their data and the tool support with them when they change organizations or jobs. A tool that supports developer improvement that cannot follow the developer as they move is not going to help those developers very much.

## **3.3 Leap toolkit: a reference implementation of the LEAP philosophy**

The Leap toolkit incorporates three main threads of research, PSP, FTR, and measurement dysfunction.

### **3.3.1 Support for personal process improvement**

The Leap toolkit is based strongly upon the PSP. The Leap toolkit uses the three primary data types, defects, size, and time, from the PSP. However, unlike the PSP, developers are able to choose what types of data to collect to help them meet their process improvement goals. If the developer is just interested in improving their estimation ability, they can record the size of their



projects and the amount of time it takes them to complete them. The Leap toolkit will provide the developer with different time estimation tools.

If the developer wants to prevent defects then they could just record their defects and not worry about size or time. The Leap toolkit will analyze their defect data and provide them insight into which defects occur most often and the developer can generate checklists that help them find those defects.

### **3.3.2 Support for Review**

From FTR, I took the idea of supporting multiple developers reviewing a work product and sharing the defects they find. The defects that others find in your work product may be more important than any of the defects you find in your own work product. By incorporating support for sharing defect data, the Leap toolkit can support reviews. In the Planning phase, review leaders can define the work product, project, defect types and checklists for the reviewers. During the preparation phase, the reviewers can use the Leap toolkit to record the defects they find in the work product. They can send their defects to the review leader who can use the Leap toolkit to combine the defects into a single list. During the review meeting the review leader can display the combined defects and each may be discussed. The author of the work product can take the combined list of defects and add it to any defects that they found. This provides the author with more data about their development process.

The Leap toolkit's flexibility allows the review leader to define their own process, defect types, and decide what review metrics they are interested in recording. The Leap toolkit will allow each reviewer to record their effort and the defects they find. The Leap toolkit can analyze the defect, time, and size data to produce reports on the defect density, defect detection rate and effectiveness of the review process.

### **3.3.3 Reducing Measurement Dysfunction**

No tool can stop measurement dysfunction. My philosophy is to acknowledge that measurement dysfunction can occur in both personal software process improvement and review. To address these issues I allow the developer full control over the data shared. The developer can decide exactly what data is shared and edit the data. This raises the measurement issues from the background to the foreground.

Even though no tool can stop measurement dysfunction, an improperly designed tool can create measurement dysfunction. If the user feels they have no control over their data, they may feel pressure to provide the “right” data and modify their behavior accordingly. This lack of control may encourage measurement dysfunction.

In combining the above three threads of research I kept the four LEAP design criteria in mind. The Leap toolkit satisfies the four design criteria. The following section describes how the Leap toolkit satisfies the design criteria.

### **3.3.4 Providing Light Weight Support**

The Leap toolkit tries to reduce the overhead of software developer improvement by automating many of the data collection process and reducing the analysis overhead by doing the difficult calculations and conversions.

The Leap toolkit, unlike the PSP or PSP Studio, does not impose any development process on the developer. If the developer wants to use the same process as PSP2.1 they may. If user does not want to have a design phase, Leap will also support that process. The user can define their own processes and Leap will support data collection and analyses based upon their processes.

The Leap toolkit also allows the user to define their own size types. This allows the user to choose a size measure that is more effective and/or convenient than lines of code used in the PSP.

### **3.3.5 Supporting Empirical Data Analysis**

The Leap toolkit allows the developer to record their effort, work product size and the defects they make while developing software. Based upon historical projects the Leap toolkit helps the developer to produce an estimate for the total amount of effort the next project will take.

### **3.3.6 Reducing Measurement Dysfunction**

The Leap toolkit stores all its data in ASCII files. This allows the developer to control the access to the data. Also, the Leap toolkit gives developers complete control over the data that they share. Developers have full control of where they save their Leap data. When users use the Leap toolkit’s email capability to share their data, the Leap toolkit asks the user what data to send. No data is shared without the user’s knowledge.

When ever the Leap toolkit is sending data over the Internet it asks what data does the developer want to send.

The toolkit also makes it very easy to edit the data before it is sent or saved. We did this for two reasons. First, if there is a data collection error then the user can edit the data to correct the error. Second, the user can edit their data before they provide it to another person. This allows the user to decide what data the other persons sees.

### **3.3.7 Providing a Portable Tool**

Since the Leap toolkit is written in Java, it can run on many different computer platforms. By using ASCII files for data storage users can easily put the files on a disk or transfer them. Both of these features allows the user to take their data and the Leap toolkit with them when they move.

## **3.4 Intended Benefits of Leap toolkit's design**

We designed the Leap toolkit to be flexible and easy to use, while supporting developer improvement. Three important benefits of the Leap toolkit's design are: (1) it prevents errors, (2) it improves time estimation, and (3) it reduces collection errors.

### **3.4.1 The Leap toolkit prevents many important classes of errors found in the PSP**

The Leap toolkit tries to address each category of data error in Disney's study. Disney's research classified the data errors into seven categories

- **Calculation Error:** The Leap toolkit does all the calculation. The user does not have to perform these calculations.
- **Blank Fields and Sequence Error:** One principle behind the Leap toolkit's design is that it should support minimal definitions. If the user does not fill in a field the Leap toolkit will do as much analysis as possible.
- **Inter- and Intra-Project Transfer Error:** The Leap toolkit handles all data transfer so the user does not have to copy data from one form to another.
- **Entry Error:** The Leap toolkit provides default values or pop-up menus for many of the important fields. This allows the user to choose from defined values reducing the chance that they will incorrectly fill in a field.
- **Impossible Values:** The Leap toolkit has a rudimentary consistency checker for some types of data. This checker indicates to the user when data values are "impossible".

### **3.4.2 The Leap toolkit improves estimation and planning**

The Leap toolkit is designed to improve the developer's estimation and planning skills. For estimating size, the Leap toolkit supports multiple size representations. The user may choose a size representation that best fits their development process. They can experiment with their size estimation abilities by using different sizes and seeing which is best for them. For example a developer can estimate the number of function points and methods that a project will be. When they complete the project they can see which estimate was more accurate.

For time estimation based upon size estimate, the Leap toolkit supports multiple estimation models. The user can choose between averages, linear regression, exponential regression, power regression and logarithmic regression. The Leap toolkit allows the user to use their planned size values or their actual size values when making an time estimate. This flexibility allows the developer to find their best method of time estimation.

The Leap toolkit also allows the user to filter their data. This allows the user to match their historical data to the current project. By matching similar projects the user's estimates should be more accurate.

### **3.4.3 The Leap toolkit reduces collection stage errors**

Automated support for entry removes simple data entry error. For example the user does not have to write down the time that they start working. This reduces the chance that they make a mistake. Also the Leap toolkit displays the current elapsed time. This feedback allows the user to check and see if the Leap toolkit is accurately recording what is happening.

Since the Leap toolkit lowers the user's overhead, it should reduce collection stage errors. The user is more likely to collect accurate data if it is easy to collect. High overhead will cause the user to not bother collecting data. Also the ease of analysis shows the user the benefit of accurate collection of data. This should motivate them to collect good data.

The next chapter discusses how I plan to evaluate these three benefits of the Leap toolkit.

# Chapter 4

## Evaluation

*There's a large journey to be taken, of many trials.* - Joseph Campbell

The main thesis of this work is LEAP provides a more accurate and effective way for developers to collect and analyze their software engineering data than methods designed for manual enactment. To evaluate this thesis I will deconstruct it into three claims based upon the three intended benefits of the Leap toolkit.

- The Leap toolkit is able to prevent many important errors in individual software engineering data collection and analysis from occurring.
- The Leap toolkit implements an approach to individual software engineering data collection and analysis that requires automated support and is not amenable to manual enactment. As a result, it enables more sophisticated approaches to data collection and analysis than is possible in a manual setting.
- The Leap Toolkit reduces the level of collection stage errors by reducing the overhead associated with collection and by mechanisms that support privacy.

The next section detail each of these claims.

### **4.1 Claim #1: Preventing important classes of errors**

To evaluate this claim I will discuss how the design and implementation of the Leap toolkit addresses each of Disney's error categories: calculation error, blank fields, inter-project transfer error, entry error, intra-project transfer error, impossible values, and sequence errors. Providing

suitable automated support should reduce these errors. Relaxing some of the constraints on the user will also remove some of these errors. Section 3.4.1 is a brief example of how I intend to address each error category.

## **4.2 Claim #2: Sophisticated approaches to data analysis**

To evaluate this claim I will use the Leap toolkit to conduct an experiment to evaluate 14 different quantitative estimation processes and the developer's estimation process and determine if there is any significant difference between the estimation methods. This experiment requires the Leap toolkit's automation to make the data collection and analysis possible.

### **4.2.1 Experimental environment**

The experiment will be performed in a student environment in the Introduction to Reflective Software Engineering course at the University of Hawaii Manoa. The students in the class will be developing 10 (software) projects and recording their software processes in the Leap toolkit. By reflecting on their experiences and the data they collect about their software development processes they should learn how to improve their development processes. During the development process they will be asked to estimate the size and amount of effort each of each software project. The Leap toolkit provides an automated tool for looking at historical development data and deriving effort estimations based upon historical size and effort data. The Leap Time estimation tool provides students with many different effort estimates based upon the students' historical data. The student can use these estimates to make their own estimate of how long the project will be.

### **4.2.2 Experimental variables**

#### **4.2.2.1 Independent and Dependent variables**

Since the objective is to evaluate the different quantitative time estimation methods, the independent variable will be the estimation technique. This means that there is one independent variable that can take on 14 different values: the 12 method values, the PSP value and the student's own estimate. The accuracy of a estimation method applies to the actual estimate itself, the mean value of all the estimates, and to the standard deviation of all the estimates. Therefore, there are three dependent variables in this experiment. The first two dependent variables are for the class as a whole. The third dependent variable is for each individual student's estimate. The relative

prediction error will be calculated for both the mean and the standard deviation for the entire class. The following two dependent variables will be calculated for each of the 14 different estimation methods for the entire class:

Mean prediction error=  $|\text{estimation mean} - \text{actual mean}|/\text{actual mean}$

Standard deviation prediction error=  $|\text{estimation std} - \text{actual std}|/\text{actual std}$

For each individual student the following dependent variable will be calculated for each of the 14 different estimation methods.

Relative Predicted Error=  $|\text{estimated time} - \text{actual time}|/\text{actual time}$

These measures cannot be measured until after the task has been completed. The different estimates must be calculated and chosen for each project before the project is started. The Leap toolkit will provide me 13 of the 14 estimates automatically. The students will record their estimate and the method(s) they use to obtain their estimate.

#### **4.2.2.2 Blocking variable**

Since the purpose of the experiment is to determine the effect of the estimation method on the prediction error and different projects may have an effect on the prediction error I will use a blocking variable to account for this effect. The reason that the different projects may have an effect on the prediction error is that it may be easier to estimate the size of some projects than others. To distinguish between the effects of the different projects and the effects of the estimation methods I have added a blocking variable associated with the project number.

#### **4.2.3 The Design**

The design for this experiment has two parts, the class as a whole and each individual student. At the beginning of every project the student will develop a planned size and planned effort for the project. After the third project, Leap will estimate the effort according to the different alternatives (treatment, alt 1 - alt 13). The student will produce the 14th estimate. During the project the students will record the amount of effort in Leap. After the project is finished, the student will measure the actual size of the project and total the actual amount of effort in Leap. After all the projects are finished, I will calculate the relative prediction error for the mean and standard deviation for each of the fourteen estimation methods. I will calculate these values for the entire class and for each individual student. We cannot use the data from the first three projects since the quantitative

estimation methods require at least three data points. With ten projects we will still have enough data points to evaluate the different estimation methods. Since the estimates are independently generated except for the students' estimate there is no problem with the order of the alternatives.

#### 4.2.4 Analysis

I am using the following relationship to model the experiment  $y_{ij} = \mu + t_i + \beta_j + e_{ij}$  where:

- $y_{ij}$  = the relative prediction error for alternative i
- $\mu$  = the overall mean
- $t_i$  = the effect of the ith treatment (estimation method)
- $\beta_j$  = the effect of the jth block (project)
- $e_{ij}$  = residual for the ith and jth treatment.

This model can be analyzed with standard analysis of variance (ANOVA) procedures with the null hypothesis:

$$H_0 : \mu_1 = \mu_2 = \mu_3 = \dots = \mu_{14} \quad \text{where} \quad \mu_i = \mu + t_i; i = \{1, 2, 3, \dots, 14\}.$$

The null hypothesis states that there is no effect of estimation methods on the prediction error. For the general comparison of the 14 methods the entire class' data will be compared. The mean prediction error and standard deviation prediction error will be calculated for each method using the entire class' data.

Mean prediction error = | Class' estimation mean - Class' actual mean | / Class' actual mean

Standard deviation prediction error = | Class' estimation std. dev. - Class' actual std. dev. | / Class' actual std. dev.

The null hypothesis can be tested for the entire class. Rejecting the null hypothesis only means that there is a difference between the accuracy of the estimation methods it does not indicate which estimation technique is more accurate. If the null hypothesis is rejected I will use the Least Significant Method to distinguish between the different alternative estimation methods. For each student I will consider the relative predicted error only. The equation for relative predicted error is



$$\text{Relative Predicted error} = \frac{|\text{estimated time} - \text{actual time}|}{\text{actual time}}$$

I will perform similar calculations and ANOVA to determine if there is an individual difference in estimation methods.

#### 4.2.5 Example Data

Table 4.1 shows some example data from my own Java development experience. This data is from a pilot study that I conducted this spring and summer. I have recorded the planned size and effort for over 20 projects beginning in December 1997. This data reflects the ten most recent projects that I have data for. I treated these ten projects just like the projects the students in the class will. Leap generated the 13 quantitative estimates and I provided the student's estimate. Since it is a single subject's data I can only calculate the relative predicted error for this data.

Table 4.1: Example Project Data.

Method	Project 4	Project 5	Project 6	Project 7	Project 8	Project 9	Project 10	Mean	Std. Dev.
APL	263	1150	386	63	48	60	151	303.00	393.84
AAL	191	763	217	38	30	37	160	205.14	258.35
LPL	315	717	509	0*	0*	0*	109	235.71	287.33
LAL	186	268	183	0*	0*	9	72	102.57	109.82
EPL	394	6634	234	118	116	99	102	1099.57	2242.82
EAL	176	321	157	139	136	112	99	162.86	74.35
APM	319	1346	405	65	101	70	174	354.29	456.16
AAM	329	1336	343	55	72	50	126	330.14	460.87
LPM	0*	179	484	0*	0*	0*	145	115.43	179.84
LAM	136	387	318	18	42	14	111	146.57	149.23
EPM	42	237	227	123	132	103	109	139.00	69.83
EAM	124	747	186	144	143	114	105	223.29	232.45
PSP	191	763	509	38	30	37**	109	239.57	286.23
Student	240	837	265	93	56	37	141	238.42	227.94
Actual	198	3047	380	139	42	26	248	582.86	1093.39

Based upon the above data I calculated the relative predicted error for all 14 estimation methods and all 7 projects. Table 4.2 shows the values for the relative predicted error.

The analysis of variance for the relative predictive error data is shown in Table 4.3.

Table 4.3 shows that the estimation method does not play a significant role in the relative prediction error. I cannot reject the null hypothesis. The data does suggest that the blocking factor, the project, does play a significant role in the prediction error. This implies that the ability to

Table 4.2: Example Predicted Error.

Method	Project 4	Project 5	Project 6	Project 7	Project 8	Project 9	Project 10
APL	0.3282	0.6226	0.0158	0.5468	0.1429	1.3077	0.3911
AAL	0.0354	0.7496	0.4289	0.7266	0.2857	0.4231	0.3548
LPL	0.5909	0.7647	0.3395	1.0000	1.0000	1.0000	0.5605
LAL	0.0606	0.9120	0.5184	1.0000	1.0000	0.6538	0.7097
EPL	0.9899	1.1772	0.3842	0.1511	1.7619	2.8077	0.5887
EAL	0.1111	0.8947	0.5868	0.0000	2.2381	3.3077	0.6008
APM	0.6111	0.5583	0.0658	0.5324	1.4048	1.6923	0.2984
AAM	0.6616	0.5615	0.0974	0.6043	0.7143	0.9231	0.4919
LPM	1.0000	0.9413	0.2737	1.0000	1.0000	1.0000	0.4153
LAM	0.3131	0.8730	0.1632	0.8705	0.0000	0.4615	0.5524
EPM	0.7879	0.9222	0.4026	0.1151	2.1429	2.9615	0.5605
EAM	0.3737	0.7548	0.5105	0.0360	2.4048	3.3846	0.5766
PSP	0.0354	0.7496	0.3395	0.7266	0.2857	0.4231	0.5605
Student	0.2121	0.7253	0.3026	0.3309	0.3333	0.4231	0.4315

Table 4.3: Example ANOVA for Predicted Error.

Source of variance	SS	Df	MS	$F_0$	p-value
Treatment (estimation method)	7.54237769	13	0.58018289	2.03137219	0.07729167
Block (project)	14.2348694	6	2.37247824	8.30666732	0.00634850
Error	22.2776831	78	0.28561132		
Total	44.05493933918	98			

estimate the size of the project is very important. This result supports the results from a study on the effects of PSP training[27]. Prechelt found engineers trained in the PSP could accurately estimate their productivity rate, but not the total size of the project or the amount of time it would take them.

### **4.3 Claim #3: Reduces collection stage errors**

To investigate this claim I will conduct a case study of the students in ICS 613. This is the most difficult part of my evaluation. Getting at the collection stage errors is very difficult since direct observation of all the students is impossible. The purpose of this case study is to determine the level of data collection errors for the students in the experiment. I will look for indications of collection errors.

#### **4.3.1 Case Study Method**

To find indications of collection errors I will conduct four surveys and analyze the students' raw data. The answers to the surveys and the analysis of the students' data will allow me to tease out suspicious data. All the students in the class will be asked to fill out the four surveys and provide me with access to their raw data. Only students that consent will be asked to fill out the surveys. If any of the students drop the course or want to quit the study their surveys will be ignored.

##### **4.3.1.1 Ensuring Anonymity**

To ensure anonymity of the surveys we will have each student place their survey in an envelope, seal and sign the envelope. This will ensure to the students that we cannot look at the surveys before the end of the class. After the grades have been turned in I will open the envelopes and randomly give the students code numbers. These numbers will be used to match up the survey results and the student's raw data. No where in the raw data or the surveys do we ask for any identifying data. Before any results are published I will ensure that any identifying material is removed.

#### **4.3.2 Data Collection**

There are two main data collection methods of this case study: the student's raw data analysis and the student surveys.

#### **4.3.2.1 Student's raw data**

As a part of ICS613 each student will turn in their Leap data for each of the class' 10 projects. I have gotten consent from each of the students to have access to their raw data. I will use the Leap toolkit to help analyze the data.

**4.3.2.1.1 Indirect Collection Error Evidence** I will look for patterns in the data that are suspicious. Some suspicious trends are having time data that is in increments of five or ten minutes or having times that start or end on the hour. The Leap toolkit records times in increments of seconds so it is extremely unlikely that anyone could work in exact increments of five or ten minutes. Such patterns indicates that the user is editing their data. This is just an indirect indication that the data does not accurately reflect their actual development.

**4.3.2.1.2 Direct Collection Error Evidence** A more direct indication of collection errors is discrepancies between the time and defect data. If the student records large amounts of test time with very few defects, this indicates that they are not recording all the defects that they had to fix during the test phase. The students are supposed to record the amount of time it takes them to fix each error. If the total fix time for all the recorded defects is substantially less than or greater than the total time spent during test then this indicates that they did not collect accurate data.

#### **4.3.2.2 Surveys**

Portions of the surveys have been used and validated at other institutions. However, we were unable to find validated surveys that ask all the questions that we want answered so portions of the surveys have not been validated externally. If these surveys reveal interesting data, we may then start another research project to validate these instruments for use in other organizations.

Each student that is in the class will be asked to fill out the surveys as they turn in their assignments. In the class the students have an interview with Dr. Johnson when they turn in each programming assignment. During the interview the student turns in their completed project and their Leap data. The focus of these interviews is on the student's performance of the assignment. I will not be collecting data about the interview, but I will get a copy of all the student's Leap data. After their interview with Dr. Johnson each student will be asked to fill out the appropriate survey.

The proposed survey schedule is

Table 4.4: Proposed Study Schedule.

<b>Task</b>	<b>Milestone</b>
Conduct 1st Survey of students	04 Oct 99
Conduct 2nd Survey of students	01 Nov 99
Conduct 3rd Survey of students	22 Nov 99
Conduct 4th Survey of students	06 Dec 99
Conduct student interviews	06 - 17 Dec 99

#### **4.3.2.3 Leap Survey #1**

This survey is intended to be a baseline survey to gather information about the students and gather initial use of the time recording tools of the Leap toolkit.

**4.3.2.3.1 Topics** This survey focuses on the student's programming experience, their experiences with time collection and any issues they have with the Leap toolkit. I will compare the student's programming experience to their feelings of pressure and detected data collection error rate.

**4.3.2.3.2 Summary of Questions for Survey #1** The first section of the survey asks about the student's programming experience. The second section asks about their knowledge of software engineering principles. Many of these principles will be briefly covered during the class. The third section asks the students about their time data collection experience. I'm trying to find out which data collection tool they use more often. The fourth section asks the students to list three negative aspects about the toolkit and three positive aspects of the toolkit. See A for the actual survey that was given to the students.

#### **4.3.2.4 Leap Survey #2**

The second survey focuses on usability issues of the Leap toolkit, the student's use of the time recording tools and time estimation. At this point in the course they will have been taught how to use the time estimation and will have used it for a few projects.

**4.3.2.4.1 Topics** The usability portion of the survey looks at the Human Computer Interface issue in the Leap toolkit. I will compare their reported easy of use with indications of collection errors. The second time survey will allow me to see if their perception or use of the time recording tools is changing over time.

**4.3.2.4.2 Summary of Questions for Survey #2** The first section of the survey asks about the Leap toolkit's usability. The second section asks the students about their time data collection experience. I'm trying to find out which data collection tool they use more often. The third section asks them about their time estimation experience. I'm trying to find out if making a plan affected their perception of the project. The fourth section asks the students to list three negative aspects about the toolkit and three positive aspects of the toolkit. See A for the actual survey that will be given to the students.

#### **4.3.2.5 Leap Survey #3**

Survey #3 repeats the time collection, and time estimation surveys from #1 and #2. In addition it asks about their defect collection experience.

**4.3.2.5.1 Topics** This survey focuses on data collection and analysis. It looks at time and defect data collection and time estimation.

**4.3.2.5.2 Summary of Questions for Survey #3** The first section of the survey asks the students about their time data collection experience. The second section asks them about their time estimation experience. The third section asks the students about their defect collection experiences. See A for the actual survey that will be given to the students.

#### **4.3.2.6 Leap Survey #4**

In the last survey I again ask the students to fill out a usability survey. I want to learn how their perceptions have changed over the semester.

**4.3.2.6.1 Topics** This survey focuses on the students' perception of the Leap toolkit.

**4.3.2.6.2 Summary of Questions for Survey #4** The first section of the survey asks about the Leap toolkit's usability. The second section asks them about their perceptions of the Leap toolkit. The third section asks the students about any lessons they may have learned from using the Leap toolkit. See A for the actual survey that will be given to the students.

### **4.3.3 Possible Interviews with students**

In addition to the surveys and raw data, I will conduct interviews with any students who are willing at the end of the course. During this interview I will ask the students about data collection issues and the overhead of using the Leap toolkit for improvement. I will also ask them about what they learned about their own development process. These interviews will support the data collected in the surveys and the raw Leap data.

# Chapter 5

## Time line

Table 5.1: Proposed Research Time line.

<b>Task</b>	<b>Milestone</b>
Develop Survey and interview questions	25 Sep 99
Form Committee	30 Sept 99
Conduct 1st Survey of students	04 Oct 99
Conduct 2nd Survey of students	01 Nov 99
Proposal to committee	29 Oct 99
Meetings with committee	08 - 12 Nov 99
Defend Proposal	19 Nov 99
Conduct 3rd Survey of students	22 Nov 99
Conduct 4th Survey of students	06 Dec 99
Conduct student interviews	06 - 17 Dec 99
Introduction, Related Work, Leap and Experimental Design Chapters done	31 Dec 99
Estimation Data analysis complete	31 Jan 00
Survey Data Analysis complete	31 Jan 00
Interview Data Analysis complete	14 Feb 00
Results Chapter Done	24 Feb 00
Dissertation to Committee	24 Feb 00
Dissertation Defense	16 Mar 00
Dissertation to Grad. Division	7 April 00



## **Appendix A**

# **Leap Evaluation Surveys**

## Leap User Survey #1

### **Instructions**

Please answer all the questions to the best of your ability. Do not put your name on this survey. When you are finished filling out the survey place it in the envelope provided. Seal the envelope and sign your name across the seal. This will ensure that no one tampers with your survey. The envelopes will not be opened until after the grades for this class are turned in.

### **Personal Programming Experience**

Please answer the following questions about your programming experience. Give your best answer. Your answers do not have to be exact.

1. Roughly how many years of programming experience do you have? \_\_\_\_\_ (years)
2. Roughly how many years of *Java* programming experience do you have? \_\_\_\_\_ (years)
3. Roughly how many total thousands of lines of code (KLOC) have you written in any language?  
\_\_\_\_\_ (KLOC)
4. Roughly how many KLOC of *Java* have you written? \_\_\_\_\_ (KLOC)
5. Roughly how many different programming languages have you used? \_\_\_\_\_ (languages)
6. Roughly how big (in KLOC) is the biggest program (or your part of a program) that you have written?  
\_\_\_\_\_ (KLOC)
7. Roughly how big (in KLOC) is the biggest Java program (or your part of a program) that you have written?  
\_\_\_\_\_ (KLOC)
8. Roughly how many graduate level ICS classes have you taken? \_\_\_\_\_ (classes)
9. Roughly how many years of paid professional programming experience do you have? \_\_\_\_\_ (years)
10. Roughly how many different paid professional programming jobs have you had? \_\_\_\_\_ (jobs)

### **Software Engineering Experience and Attitudes**

Please circle the number that most closely matches your feelings about the following statements. If the statement does not apply to you circle NA.

	Strongly Disagree					Strongly Agree	
	1	2	3	4	5		NA
I am familiar with many different software development processes.	1	2	3	4	5		NA
I have used many different software development processes.	1	2	3	4	5		NA
I understand Object Oriented Programming.	1	2	3	4	5		NA
I can explain the benefits of Object Oriented Programming to other programmers.	1	2	3	4	5		NA
I know what the Personal Software Process is.	1	2	3	4	5		NA
I have used the Personal Software Process.	1	2	3	4	5		NA
I am aware of my own software development process.	1	2	3	4	5		NA
I am a good software developer.	1	2	3	4	5		NA
I want to be a better software developer.	1	2	3	4	5		NA
I can explain what it means to be a good software developer.	1	2	3	4	5		NA

Figure A.1: Leap survey #1 page 1

### Leap Usage – Time Collection

Please circle the answer that most closely matches your use of the features in Leap. Choose NA if the question does not apply to you.

Questions	Answers					
	Io (single line entry tool)			Naia (time table)		
Which time entry tool do you use most often?						
How often do you use Io (single line) to record your time?	Never (0%)	Less than half the time (1 – 39%)	About half the time (40 - 60%)	More than half the time (61 – 99%)	All the time (100%)	NA
How often do you use Naia (table) to record your time?	Never (0%)	Less than half the time (1 – 39%)	About half the time (40 - 60%)	More than half the time (61 – 99%)	All the time (100%)	NA
I use Naia to edit my time data.	Never (0%)	Less than half the time (1 – 39%)	About half the time (40 - 60%)	More than half the time (61 – 99%)	All the time (100%)	NA
I prefer to use Io for time recording.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I prefer to use Naia for time recording.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My Leap data accurately reflects what really happened	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

Please list the most negative aspect(s) of the Leap toolkit, in your opinion.

1.

2.

3.

Please list the most positive aspect(s) of the Leap toolkit, in your opinion.

1.

2.

3.

If you would like to provide any additional comments about the Leap toolkit, please note them on the rest of this page and the back of this page. Thank you for taking the time to fill out this survey. Please place it in the envelope, seal and sign the envelope.

Figure A.2: Leap survey #1 page 2

## Leap User Survey #2

### **Leap Usability survey**

Please circle the number that most closely matches your feelings about the following statements. If the statement does not apply to you circle NA.

	Strongly Disagree					Strongly Agree	
	1	2	3	4	5		NA
Overall, I am satisfied with how easy it is to use Leap.							
It is simple to use Leap.							
I can effectively complete the Leap portions of my assignments.							
I can efficiently complete the Leap portions of my assignments							
It is easy to learn to use Leap.							
I believe I became productive quickly using Leap.							
Leap gives error messages that clearly tell me how to fix problems.							
Whenever I make a mistake using Leap, I recover easily and quickly.							
The information (such as on-screen messages, and other documentation) provided with Leap are clear.							
It is easy to find the information I need.							
The information provided with Leap is easy to understand.							
The information is effective in helping me complete the tasks and assignments.							
The organization of information on Leap screens is clear.							
The interface of Leap is pleasant.							
I like using the interface of Leap.							
Leap has all the functions and capabilities I expect it to have.							
Overall, I am satisfied with Leap.							

Figure A.3: Leap survey #2 page 1

**Leap Usage – Time Collection**

Please circle the answer that most closely matches your use of the features in Leap. Choose NA if the question does not apply to you.

Questions	Answers					
	Io (single line entry tool)			Naia (time table)		
Which time entry tool do you use most often?						
How often do you use Io (single line) to record your time?	Never (0%)	Less than half the time (1 – 39%)	About half the time (40 - 60%)	More than half the time (61 – 99%)	All the time (100%)	NA
How often do you use Naia (table) to record your time?	Never (0%)	Less than half the time (1 – 39%)	About half the time (40 - 60%)	More than half the time (61 – 99%)	All the time (100%)	NA
I use Naia to edit my time data.	Never (0%)	Less than half the time (1 – 39%)	About half the time (40 - 60%)	More than half the time (61 – 99%)	All the time (100%)	NA
I prefer to use Io for time recording.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I prefer to use Naia for time recording.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My time data gives me valuable insights into my strengths as a programmer	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My Leap data accurately reflects what really happened	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My time data gives me valuable insights into my weaknesses as a programmer	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

**Leap Usage – Time Estimation**

Please circle the answer that most closely matches your use of the features in Leap. Choose NA if the question does not apply to you.

Questions	Answers					
I am aware of my time estimate while I do my assignment.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel comfortable with my time estimates.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel comfortable with my ability to estimate project size.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel pressure to make my actual effort match my estimated effort.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel pressure to make my actual project size match my estimated project size.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel pressure to turn in data where my actual size and time data matches my estimates.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Time pressure reduces the quality of the data I collect in Leap.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

Figure A.4: Leap survey #2 page 2

Please list the most negative aspect(s) of the Leap toolkit, in your opinion.

1.

2.

3.

Please list the most positive aspect(s) of the Leap toolkit, in your opinion.

1.

2.

3.

What have you learned about your own software development process?

If you would like to provide any additional comments about the Leap toolkit, please note them on the rest of this page and the back of this page. Thank you for taking the time to fill out this survey.

Figure A.5: Leap survey #2 page 3

## Leap User Survey #3

### ***Leap Usage – Time Collection***

Please circle the answer that most closely matches your use of the features in Leap. Choose NA if the question does not apply to you.

Questions	Answers					
Which time entry tool do you use most often?	Io (single line entry tool)			Naia (time table)		
I prefer to use Io for time recording.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I prefer to use Naia for time recording.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I use Naia to edit my time data.	Never (0%)	Less than half the time (1 – 39%)	About half the time (40 - 60%)	More than half the time (61 – 99%)	All the time (100%)	NA
My Leap data accurately reflects what really happened	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My time data gives me valuable insights into my strengths as a programmer.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My time data gives me valuable insights into my weaknesses as a programmer	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

### ***Leap Usage – Time Estimation***

Please circle the answer that most closely matches your use of the features in Leap. Choose NA if the question does not apply to you.

Questions	Answers					
I am aware of my time estimate while I do my assignment.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel comfortable with my time estimates.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel comfortable with my ability to estimate project size.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel pressure to make my actual effort match my estimated effort.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel pressure to make my actual project size match my estimated project size.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I feel pressure to turn in data where my actual size and time data matches my estimates.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

Figure A.6: Leap survey #3 page 1

My time estimation skills are improving.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My size estimation skills are improving.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

**Leap Usage – Defect Collection**

Please circle the answer that most closely matches your feelings. Choose NA if the question does not apply to you.

Questions	Answers					
Being aware of my defects helps me avoid making them in the future.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My defect data gives me valuable insights into my strengths as a programmer.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My defect data gives me valuable insights into my weaknesses as a programmer	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Collecting my defect data is a waste of my time.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My Leap data accurately reflects what really happened	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Time pressure reduces the quality of the data I collect in Leap.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

What have you learned about your project estimation abilities?

If you would like to provide any additional comments about the Leap toolkit, please note them on the back of this page. Thank you for taking the time to fill out this survey.

Figure A.7: Leap survey #3 page 2



## Leap User Survey #4

### ***Leap Usability survey***

Please circle the number that most closely matches your feelings about the following statements. If the statement does not apply to you circle NA.

	Strongly Disagree					Strongly Agree		
	1	2	3	4	5			NA
Overall, I am satisfied with how easy it is to use Leap.								NA
It is simple to use Leap.								NA
I can effectively complete the Leap portions of my assignments.								NA
I can efficiently complete the Leap portions of my assignments								NA
It is easy to learn to use Leap.								NA
I believe I became productive quickly using Leap.								NA
Leap gives error messages that clearly tell me how to fix problems.								NA
Whenever I make a mistake using Leap, I recover easily and quickly.								NA
The information (such as on-screen messages, and other documentation) provided with Leap are clear.								NA
It is easy to find the information I need.								NA
The information provided with Leap is easy to understand.								NA
The information is effective in helping me complete the tasks and assignments.								NA
The organization of information on Leap screens is clear.								NA
The interface of Leap is pleasant.								NA
I like using the interface of Leap.								NA
Leap has all the functions and capabilities I expect it to have.								NA
Overall, I am satisfied with Leap.								NA

Figure A.8: Leap survey #4 page 1

**Perceptions of Leap**

Please circle the answer that most closely matches your feelings. Choose NA if the question does not apply to you.

Questions	Answers					
Using Leap enables me to develop software more quickly.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap improves the quality of my software.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap makes it easier for me to develop software.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap improves my software development performance.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Overall, I find using Leap to be advantageous in my software development.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap enhances my effectiveness in software development.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap gives me greater control over my work.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap increases my productivity.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap is compatible with all aspects of my software development process.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I think using Leap fits well with the way I develop software.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap fits into my work style.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I believe that Leap is cumbersome to use.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My using Leap requires a lot of mental effort.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap is often frustrating.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I believe that it is easy to get Leap to do what I want it to do.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Overall, I believe that Leap is easy to use.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I would have no difficulty telling others about the results of using Leap.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I believe I could communicate to others the consequences of using Leap.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

Figure A.9: Leap survey #4 page 2

The results of using Leap are apparent to me.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I would have difficulty explaining why using Leap may or may not be beneficial.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

**Lessons Learned**

Please circle the answer that most closely matches your feelings. Choose NA if the question does not apply to you.

Questions	Answers					
Leap has shown me valuable insights into my software development process.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My programming skills have improved since the beginning of this class.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I will continue to use Leap as a part of my software development.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Collecting data about my software development process has been a waste of my time.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
My Leap data accurately reflects what really happened	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
Using Leap has made me a better software developer.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA
I can describe my own software development process.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	NA

Please describe the most valuable insights you have learned about your software development process.

Thank you for taking the time to fill out this survey.

Figure A.10: Leap survey #4 page 3

# Bibliography

- [1] Jody Armour and Watts S. Humphrey. Software product liability. Technical Report CMU/SEI-93-TR-13, Software Engineering Institute, Carnegie Mellon University, August 1993.
- [2] Robert D. Austin. *Measuring and managing performance in organizations*. Dorset House, 1996.
- [3] Peter M. Blau. *The Dynamics of Bureaucracy: A Study of Interpersonal Relations in Two Government Agencies*. The University of Chicago Press, 2nd edition, 1963.
- [4] Anne M. Disney. Data quality problems in the Personal Software Process. M.S. thesis, University of Hawaii, August 1998.
- [5] Anne M. Disney and Philip M. Johnson. Investigating data quality problems in the PSP. In *Proceedings of the ACM SIGSOFT Sixth International Symposium on the Foundations of Software Engineering*, pages 143–152, Lake Buena Vista, FL, November 1998.
- [6] Robert H. Dunn. *Software Quality: Concepts and Plans*. Prentice Hall, 1990.
- [7] Khaled El Emam, Barry Shostak, and Nazim Madhavji. Implementing concepts from the Personal Software Process in an industrial setting. In *Proceedings of the Fourth International Conference on the Software Process*, Brighton, England, December 1996.
- [8] Michael E. Fagan. Design and code inspections to reduce errors in program development. *IBM Systems Journal*, 15(3):182–211, 1976.
- [9] Pat Ferguson, Watts S. Humphrey, Soheil Khajenoori, Susan Macke, and Annette Matvya. Introducing the Personal Software Process: Three industry cases. *IEEE Computer*, 30(5):24–31, May 1997.
- [10] The WWW formal technical review archive. <http://www.ics.hawaii.edu/johnson/FTR/>.

- [11] Robert L. Glass. *Software Runaways: Lessons Learned from Massive Software Project Failures*. Prentice Hall, 1998.
- [12] GNU emacs - GNU project - free software foundation(FSF). <http://www.gnu.org/software/emacs/emacs.html>.
- [13] Will Hayes and James W. Over. The Personal Software Process (PSP): An empirical study of the impact of PSP on individual engineers. Technical Report CMU/SEI-97-TR-001, Software Engineering Institute, Pittsburgh, PA., 1997.
- [14] Joel Henry. Personal Software Process studio. <http://www-cs.etsu.edu/softeng/psp/>, 1997.
- [15] Watts S. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley, January 1995.
- [16] Watts S. Humphrey. Using a defined and measured Personal Software Process. *IEEE Software*, 13(3):77–88, May 1996.
- [17] International Organization for Standardization. *ISO Standards Compendium - ISO 9000 Quality Management*, 7th edition, 1998.
- [18] Philip M. Johnson. An instrumented approach to improving software quality through formal technical review. In *Proceedings of the 16th International Conference on Software Engineering*, pages 113–122, Sorrento, Italy, May 1994.
- [19] Philip M. Johnson. Supporting technology transfer of formal technical review through a computer supported collaborative review system. In *Proceedings of the Fourth International Conference on Software Quality*, Reston, VA., October 1994.
- [20] Philip M. Johnson. Design for instrumentation: High quality measurement of formal technical review. *Software Quality Journal*, 1995.
- [21] Philip M. Johnson and Danu Tjahjono. Improving software quality through computer supported collaborative review. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work*, September 1993.
- [22] Philip M. Johnson, Danu Tjahjono, Dadong Wan, and Robert Brewer. Experiences with CSRS: An instrumented software review environment. In *Proceedings of the Pacific Northwest Software Quality Conference*, Portland, OR., 1993.

- [23] S. Khajenoori and I. Hirmanpour. An experiential report on the implications of the Personal Software Process for software quality improvement. In *Proceedings of the Fifth International Conference on Software Quality*, pages 303–312, October 1995.
- [24] Nancy G. Leveson and Clark S. Turner. An investigation of the Therac-25 accidents. *IEEE Computer*, 1993.
- [25] The locc system. <http://csdl.ics.hawaii.edu/Tools/LOCC/LOCC.html>.
- [26] Mark C. Paulk, Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley, 1995.
- [27] Lutz Prechelt and Barbara Ungber. A controlled experiment on the effects of psp training: Detailed description and evaluation. Technical Report 1/1999, Fakultät für Informatik Universität Karlsruhe, April 1999.
- [28] PSP resources page at the University of Karlsruhe. <http://wwwipd.ira.uka.de/~gramberg/PSP/>.
- [29] PSPtool version 0.6. <http://www.virtual.net.au/simttqc/description.html>.
- [30] M. Ramsey. Experiences teaching the Personal Software Process in academia and industry. In *Proceedings of the 1996 SEPG Conference*, 1996.
- [31] Khalid Sherdil and Nazim H. Madhavji. Human-oriented improvement in the software process. In *Proceedings of the 5th European Workshop on Software Process Technology*, October 1996.
- [32] Barry Shostak. Adapting the Personal Software Process to industry. *Software Process Newsletter #5*, Winter 1996.
- [33] Timelog. <http://www.kclee.com/clemens/java/timelog/>.
- [34] Timetracker - an x-windows timekeeper. <http://www.alvestrand.no/domen/titrax/TimeTraker.html>.
- [35] Danu Tjahjono. Evaluating the cost-effectiveness of formal technical review factors. Ph.D. Dissertation Proposal. CSDL-TR-94-07, University of Hawaii, Department of Information and Computer Sciences, 1994.
- [36] Andrew Worsley. What are the benefits of the PSP software process? [http://www3.cm.deakin.edu.au/~peter/PSP\\_data/talk.html](http://www3.cm.deakin.edu.au/~peter/PSP_data/talk.html), 1996.
- [37] XEmacs: The next generation of emacs. <http://www.xemacs.org/>.

[38] Edward Yourdon. *Structured Walkthroughs*. Prentice-Hall, 1979.