2002

Personal Software Process (PSP) Scriber

Heng-Jui Tsao

PERSONAL SOFTWARE PROCESS (PSP) SCRIBER

A Project

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirement for the Degree

Master of Science

in

Computer Science

---

by

Heng-Jui Tsao

June 2002

PERSONAL SOFTWARE PROCESS (PSP) SCRIBER

A Project

Presented to the

Faculty of

California State University,

San Bernardino

by

Heng-Jui Tsao

June 2002

[redacted] Approved by:

[redacted] 10 Jun 2002

Dr. Arturo I Concepcion, Chair, Computer Science Date

[redacted]

Dr. Josephine Mendoza

[redacted]

Dr. Kay R. Zemoudeh

ABSTRACT

The goal of this project is to provide an on-line computer aided software engineering tool for supporting Personal Software Process strategy. The project is called PSP Scriber. PSP Scriber is an automatic Web-based system that allows a user to learn the PSP discipline of software engineering.

PSP Scriber contains all of the scripts, forms, templates, and standards that PSP requires from assignment kit 1 through assignment kit 11 defined by Watts S. Humphrey in his book "A Discipline for Software Engineering". Each assignment kit contains a complete set of forms that the user must fill up and create a history of data needed to predict project size and effort that are used to plan the entire software development. PSP Scriber provides cross-referencing of data between each form to avoid entering wrong data.

The PSP Scriber uses HTML and Java Server Page to generate basic output format. Also, PSP Scriber uses Java Bean and Java Applet for performing calculation, data update, and database connection. With the Oracle database, PSP Scriber can store the input data from the users.

TABLE OF CONTENTS

CHAPTER TWO : SOFTWARE DESIGN

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

SOFTWARE REQUIREMENTS SPECIFICATION

1.1 Introduction

1.1.1 Purpose

Personal Software Process (PSP) Scriber is a Web-based software engineering tool designed to implement an automatic system for performing Personal Software Process (PSP). Geared towards the individual software developer, it encapsulates a Personal Software Process (PSP) Strategy, as defined by Watts S. Humphrey in his book "*A Discipline for Software Engineering*". The basis of this Personal Software Process strategy is a set of tools to facilitate collection and analysis of development data. By analyzing the collected data, the developer can make informed, accurate decisions about their individual development effort. If an entire team uses the Personal Software Process (PSP) strategy, software management can make informed decisions about the entire project, helping to ensure software that is delivered on time and on budget.

PSP Scriber is the Master Project for Mr. Heng-Jui Tsao to fulfill the requirements for Master of Computer Science degree.

1.1.2 Scope

PSP Scriber serves the following basic needs:

- Delivers forms required for all Personal Software Process Kits.
- Provides scripts documenting the use of all Personal Software Process forms.
- Provides example forms, complete with sample data.
- Accepts data from users and save that data in a remote database.
- Calculates and analyses data which have been entered by the software engineer.
- Provide subsequent estimation techniques for Recursive Multi-Threaded Software Project Management Tool. The following diagram shows how Personal Software Process will integrate with Recursive Multi-Threaded Software Project Management Tool,

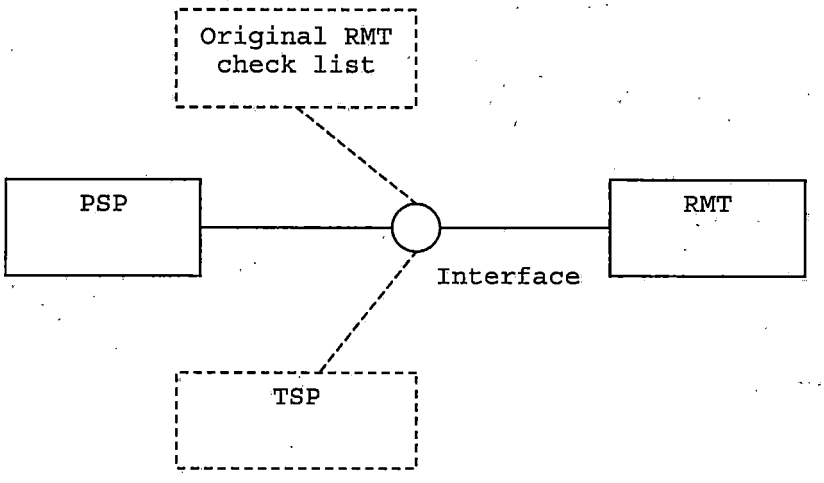


Figure 1. Personal Software Process Integrate with Recursive Multi-Threaded

1.1.3 Definitions, Acronyms and Abbreviations

- Applet

  An applet is a small program that can be sent along with a Web page to a user. Java applets can perform interactive animations, immediate calculations, or other simple tasks without having to send a user request back to the server.

- Browser

  A graphical display application used to display world wide web pages. Basic functions include navigation and printing.

- DBMS

  A DBMS (database management system), sometimes just called a database manager, is a program that lets one or more computer users create, manage and access data in a database.

- GUI - Graphical User Interface

  Interface system that relies on graphics as well as text to display and convey information to the user.

- HTML v. 3.2 - Hypertext Markup Language

  HTML (Hypertext Markup Language) is the set of markup symbols or codes inserted in a file intended for display on a World Wide Web browser. The markup tells the Web browser how to display a Web page's words and images for the user.

  HTML v. 3.2 is the first version of HTML to support tables.

- Hyperlink

  A highlighted portion of text that causes a browser to take some action such as displaying another page or graphic when clicked.

- Java

  Java is a programming language expressly designed for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces a completely object-oriented view of programming. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build small application modules or applets for use as part of a Web page. Applets make it possible for a Web page user to interact with the page.

- Java Bean

  The Java Bean technology is a significant extension and enhancement for the Java language. This technology enabled programmers to rapidly build applications by assembling objects and testing them during design time.

- JDBC

  JDBC (Java Database Connectivity) is an application program interface (API) specification for connecting programs written in Java to the data in popular databases.

- JSP

  JSP (Java Server Page) is a technology for controlling the content or appearance of Web pages through the use of servlets. It enables one to mix regular, static HTML with dynamically generated content from servlets. JSP calls a Java program that is executed by the Web server

- Kits

  PSP Scriber bundles forms, scripts and examples into logical units called kits. For example, the set of forms, scripts and examples for PSP0 is referred to as Kit 1.

- Personal Software Process - PSP

  The Personal Software Process (PSP) is a self-improvement process designed to help developers control, manage and improve the way they work. It is a structured framework of forms, guidelines and procedures for developing software. The whole PSP progression is shown on Figure 2.

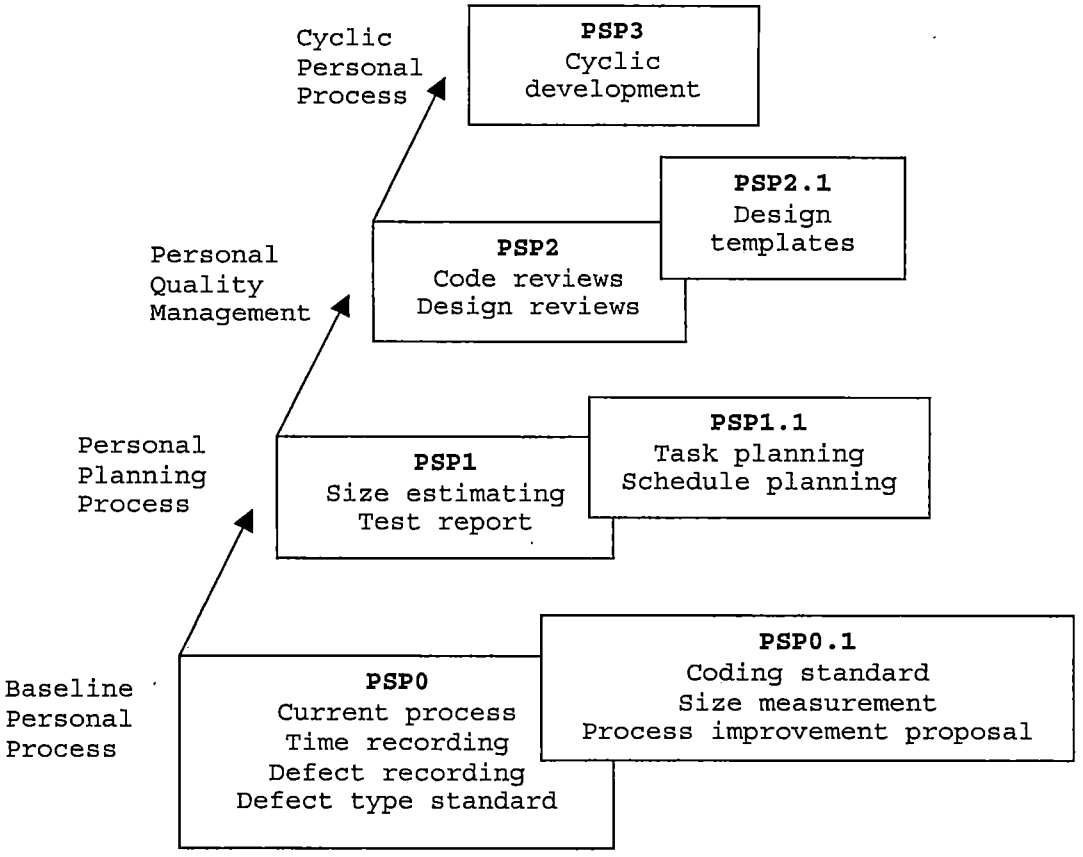


Figure 2. The Personal Software Process Evolution (Adapted from A Discipline for Software Engineering)

- RMT

RMT is a Recursive Multi-Threaded software life cycle model, and was a result of the master thesis of Scott Simon at California State University, San Bernardino. RMT uses a thread-based approach to organize development and to monitor progress. RMT is a milestone-based,

iterative life cycle that supports incremental and parallel development.

- Servlet

  Servlets are Java programs that run on a web server, acting as a middle layer between a request coming from a web browser or other HTTP client and database or application on the HTTP server.

- Swing

  Swing is a part of Java Foundation Classes (JFC) that implements a new set of GUI components with a pluggable look and feel. Swing is implemented in 100% pure Java, and is based on JDK 1.1 Lightweight UI Framework. The pluggable look and feel lets one design a single set of GUI components that can automatically have a look and feel of any OS platform (Windows, Solaris, Linux, Macintosh). Swing components include both 100% pure Java versions of the existing AWT components set (Button, Scrollbar, Label, etc.), plus a rich set of higher-level components (such as tree view, list box, and tabbed panes).

- TSP

  The Team Software Process provides a balanced emphasis on process, product, and team work, and it capitalizes on the broad base of industrial experience in planning and managing software projects.

- UML

  The Unified Modeling Language (UML) is the industry-standard language for specifying visualizing, constructing, and documenting the artifacts of software systems. It simplifies the complex of software design, making a "blueprint" for construction. The UML definition was led by Rational software's industry-leading methodologists: Grady Booch, Ivar Jacobson, and Jim Rumbaugh.

1.1.4 Overview

This Software Requirements Specifications will contain the following: First I give an introduction that contains an overview of the entire software requirements specification. It defines the scope and purpose of the project and the requirements document, defines terms used in the SRS, and supplies a list of references cited in the SRS. Then I describe the overall description that contains

an overview of factors that affect the product and its requirements. It describes any relationship this product has with other products and provides a high-level view of product functionality. Then I specify the product functionalities that contain a highly detailed description of the product and its functionality.

1.2 Overall Description

1.2.1 Product Perspective

The PSP process starts from PSP0, which establishes a baseline that includes some basic measurements and reporting format. The second process, PSP1 adds planning steps to PSP0. And then PSP2 adds review techniques to PSP1 to help a software engineer find defects early when they are least expensive to fix. Finally, PSP3 is the strategy to subdivide a large program into PSP2-sized pieces. Beside that, PSP Scriber changed the Object-Oriented approach and design to UML instead of some required forms (e.g. the Cyclic Summary Forms, the Issue Tracking Log, the Operational scenario Template, and the Functional Specification Template) by Humphry.

1.2.1.1 System Interface. PSP Scriber is composed of five major components: HTML forms, Java Server Pages forms,

the Java Bean-based JDBC classes, the Java based PSP applets and PSP database.

PSP Scriber relies on HTML 3.2 compliant browsers to provide navigation, display, text and image rendering, and basic print capabilities for all HTML forms. The client-side browsers will manage all GUI requirements for PSP Scriber form presentation.

The Java Server Pages forms rely on JSP 1.1, and use Tomcat 3.2 as the web server. The Java Server Pages forms will display dynamic contents of all forms PSP required.

The JDBC classes rely on the Java Bean technology to perform the connection between the PSP applets and PSP database. It will make it easy to send SQL statements to a relational database system and send back the result to PSP applets.

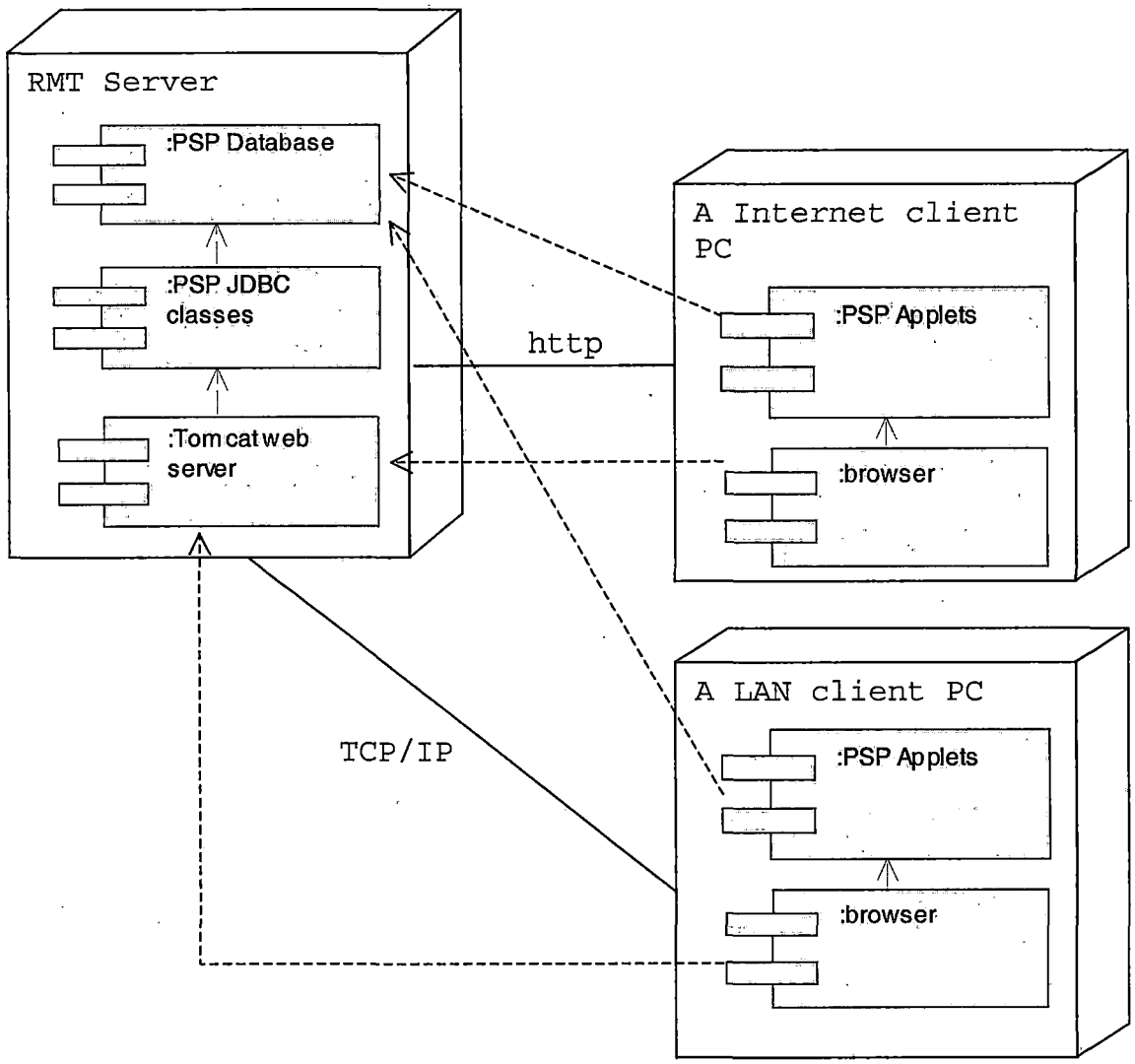


Figure 3. Component and Deployment Diagram

The PSP applet component of PSP Scriber relies on Java applet technology to provide navigation, display, calculations, text and image rendering. Additionally, the Java applet will produce PSP applet, which requires the Sun Java 2 plug in, as well as a browser that supports the

plug-in, such as Internet Explorer 4.01 or greater, or Netscape 4.0. PSP applet also requires the Swing library, and will only run on platforms that support Swing.

PSP database uses Oracle 8 as its database management system (DBMS), which will store the whole data need for HTML forms and PSP applets.

1.2.1.2 User Interfaces. PSP Scriber's target audiences are software engineers.

Accessible from any HTML 3.2 compliant browser, PSP Scriber uses HTML frames to segment the interface into two distinct sections, a contents index and a contents page.

The left frame contains the contents index page. This contents index is displayed throughout all PSP Scriber pages, and enables single-click navigation throughout the PSP Scriber site.

The right frame window will contain any page in the PSP Scriber system excluding the content index page. The welcome page is displayed upon initial entry into the system.

1.2.1.3 Hardware Interface. The client-side browser and operating system manage hardware interface issues. PSP Scriber has no known hardware related interface issues.

1.2.1.4 Software Interface. PSP Scriber relies on HTML 3.2 compliant browsers to provide navigation, display, text and image rendering, and basic print capabilities. The client-side browsers will manage all GUI requirements for PSP Scriber. PSP Scriber will run on any operating system platform containing browsers that support HTML 3.2.

The PSP applet functionality included in PSP Scriber requires the Sun Java 2 plug-in, as well as Internet Explorer version 4.01 or greater, Netscape Navigator 4.0, or an Internet Explorer compatible browser. PSP applet also requires the Swing library, and will not function on platforms that do not support Swing.

1.2.1.5 Communication Interface. PSP Scriber relies on the client browser and operating system, and the server browser, JDBC, DBMS and operating system, to manage communications issues.

1.2.1.6 Memory Constraint. PSP Scriber requires at least 45 MB of server storage space. PSP Scriber has to

meet the requirement of Internet Explorer 4.01 or greater, Netscape Navigator 4.0, or an IE compatible browser for client-side memory constraint issues.

1.2.1.7 Operations. PSP Scriber is launched by the user, typically by manually entering the URL for the PSP Scriber home page or by clicking on a link that causes one of the Scriber pages to load. The PSP system is active while the user remains within the product page set. PSP Scriber is strictly an interactive application. There are no unattended or automated processes defined as part of PSP Scriber.

All PSP Scriber data is stored as hard-coded HTML 3.2 pages. The PSP applet is stored as Java code, and is downloaded to the client and run in the client process space. The PSP user process data is stored in DBMS and available for PSP Scriber to access.

1.2.1.8 Site Adaptation Requirements. PSP Scriber has no known site adaptation requirements at this time.

1.2.2 Product Functions

PSP Scriber provides the following basic functionality:

- Supplies an overview of all PSP forms and a basic introduction to PSP
- Displays PSP forms organized by kit.
- Allows navigation between forms in a kit via web links, browser back button.
- Allows printing individual forms using the browser print function.
- Allows RMT user to access data and forms.

PSP Scriber also provides the following data entry and reporting functionalities:

- Displays PSP data entry forms - these forms allow actual data entry and editing.
- Displays PSP reports, based on data entered by user and computations performed on entered data.
- Displays an interface for navigation through the above functionality.

The following use case diagram shows the basic functionality.

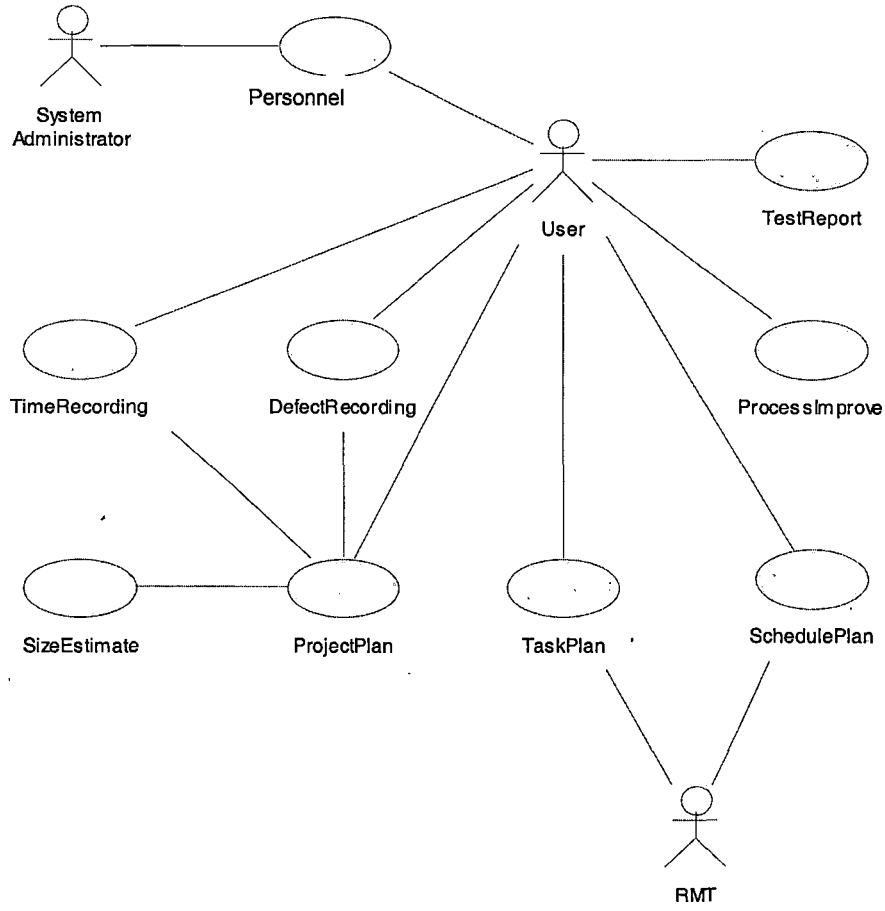


Figure 4. Use Case Diagram

1.2.3 User Characteristics

PSP Scriber is primarily designed for software engineers and managers. These users are likely to be technically competent and familiar with browser technology. No special consideration is given to documenting the use of browser technology in PSP Scriber.

1.2.4 Constraints

PSP Scriber forms and scripts are based on HTML v. 3.2, and do not use client-side executable code. Because of this, PSP Scriber forms and scripts do not require any functionality that relies on client-side executable code.

The PSP applet tool included with PSP Scriber relies heavily on Java applet technology, and the Swing library. PSP applet will run on any platform that supports Java applets (specifically, the Sun Java 2 plug-in) and Swing.

PSP Scriber relies on tables, and will only run on browsers that are HTML v. 3.2 compliant.

1.2.5 Assumptions and Dependencies

PSP Scriber assumes the structure and content of documents included in the system are stable and not subject to constant change. While the structure and content of forms can be modified, doing so will be tedious. No provision for versioning has been included in this release.

1.3 Specific Requirements

1.3.1 External Interfaces

PSP Scriber Version 1.1 will allow the user to navigate through PSP forms and scripts via the following browser related actions:

- Clicking on hyperlinks.
- Using the forward and back buttons on the browsers' navigation toolbar.
- Any additional operations supported by the browser for general HTML navigation.

PSP Scriber includes a PSP applet component, which will allow the following types of interactions:

- Navigation throughout the component via a hierarchical menu system
- Use of property sheets to segment data into logical groupings - the user will need to navigate between property pages using tabs at the top of the sheet

The source of these inputs will be the PSP Scriber user who will be navigating throughout the system in search of particular forms, scripts, instructions or examples.

The output of the system will consist on all of displayed and printed HTML pages. As mentioned, the pages will be selected by the user via browser navigational features, or applet navigational features when using PSP Applet. All pages in PSP scriber including PSP Forms, scripts, instructions, and examples and PSP Applet reports can be printed using this process.

To allow for an improved interaction with the user, all pages will be formatted in a consistent, standard fashion that closely simulates the original document. This standardization of format will bring a cohesive feel to the system as all screens and output will have a consistent, familiar layout.

1.3.2 Functions

The PSP Scriber shall perform all standard navigational functions supported by HTML and web browsers. All forms referenced from the kits and other forms will be accessible via HTML hyperlinks. Each kit within the PSP Scriber includes a hyperlink section for easy access to all forms, scripts, and references applicable to the kit.

1.3.2.1 Applet General Requirement. PSP Applet is a computational tool included with PSP Scriber. PSP Applet

will download the existing data and prompts the user to enter a minimal amount of data, computes a variety of PSP statistics (e.g. liner regression) as defined by Humphrey, and generates a printable report using Java Server Page. Also, PSP Applet will perform functions to save the data into DBMS.

1.3.2.2 Login Page. The main access to PSP Scriber is through the Welcome Page, which will provide login field for the user and allows PSP database to bring background data for that user. After the user logs in, PSP Scriber will bring it to the Welcome page.

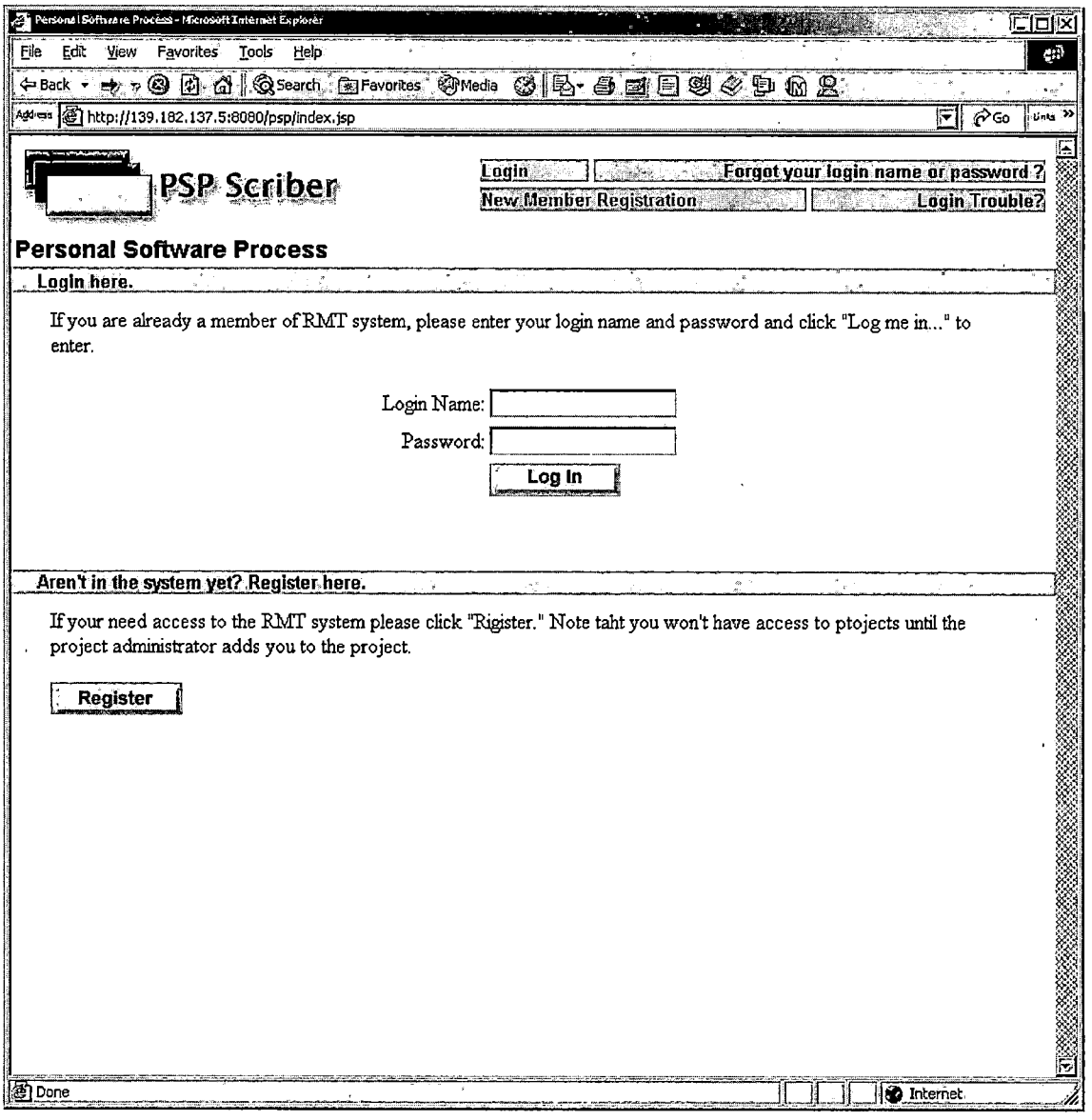


Figure 5. Screen Shot of Login Page

1.3.2.3 Welcome Page. The Welcome page introduces the background of the software engineering and the purpose of PSP Scriber. The frame in the left will be presented as an index, including home, and all the forms organized by each

kit. The user will be allowed to navigate each kit by a single click on the index. The welcome page will also be obtained by clicking home during user navigation.

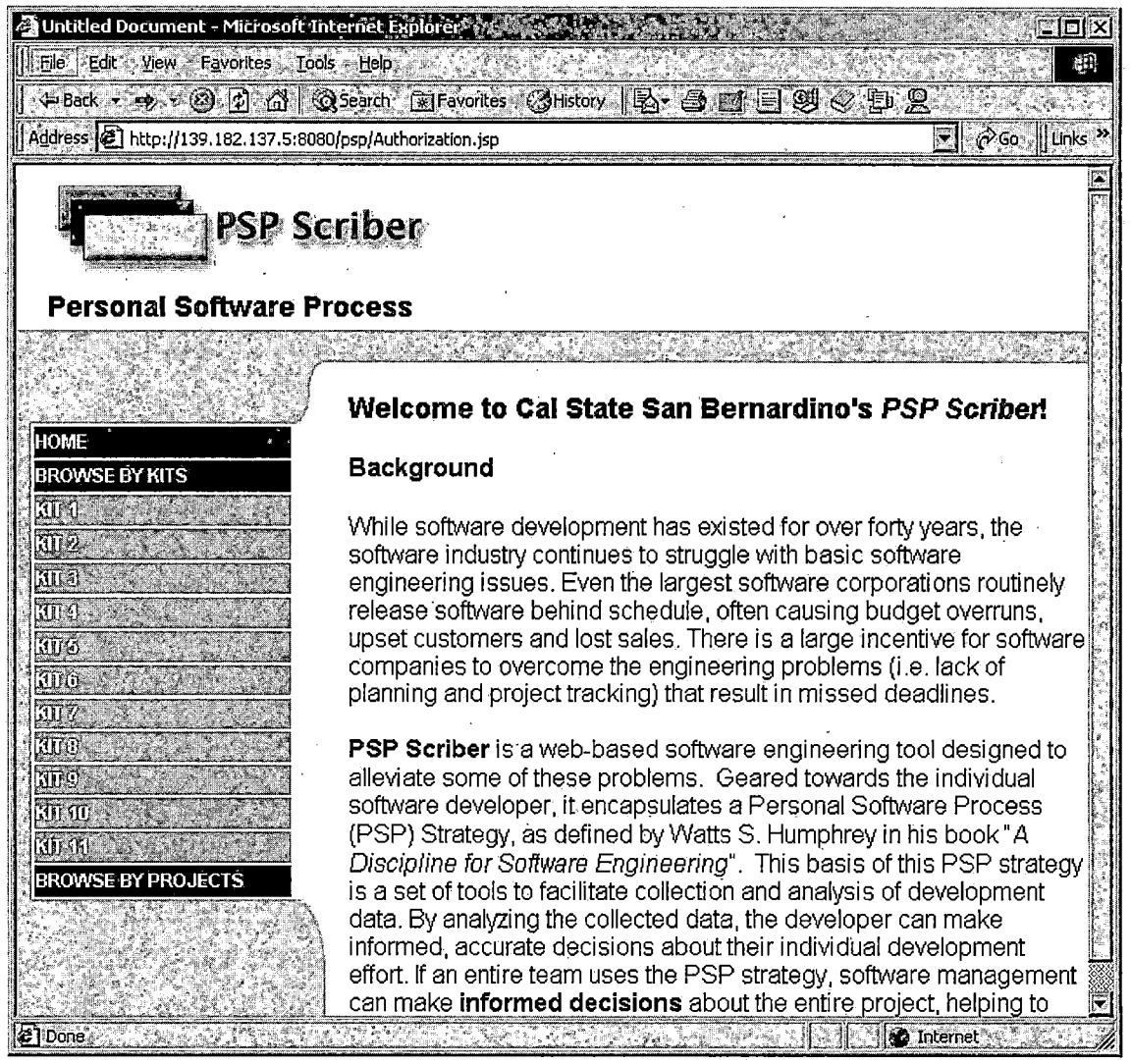


Figure 6. Screen Shot of Welcome Page

1.3.2.4 Kit 1.

1.3.2.4.1 Kit 1 Main page. The Kit 1 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

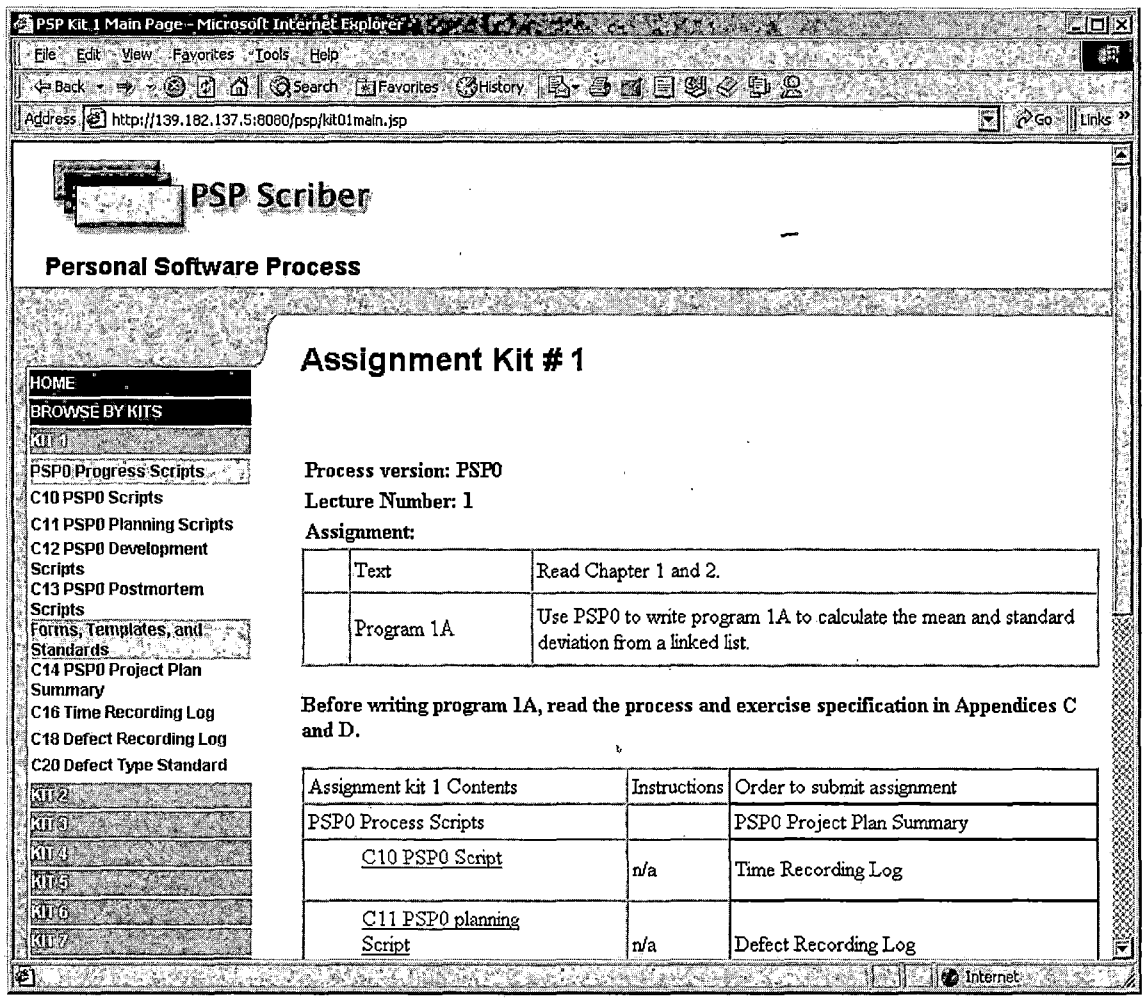


Figure 7. Screen Shot of Kit 1 Main Page

1.3.2.4.2 Kit 1 PSP0 Project Planning Summary.

The PSP0 Project Plan Summary also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the PSP0 Project Plan Summary. The user can enter data on this PSP applet. The applet will perform the calculate function (e.g. to date planning time) defined by Humphrey, and save

data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

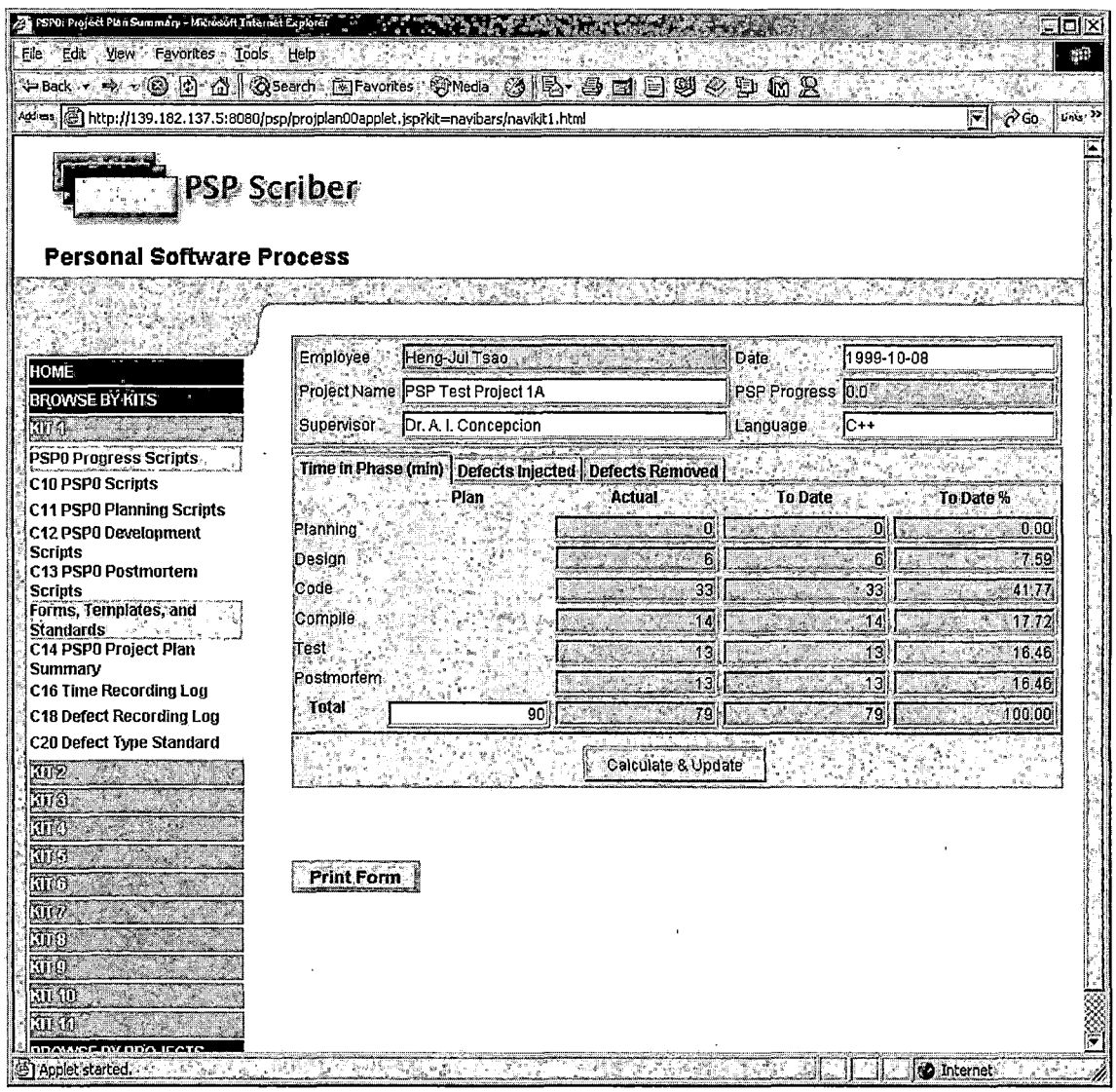
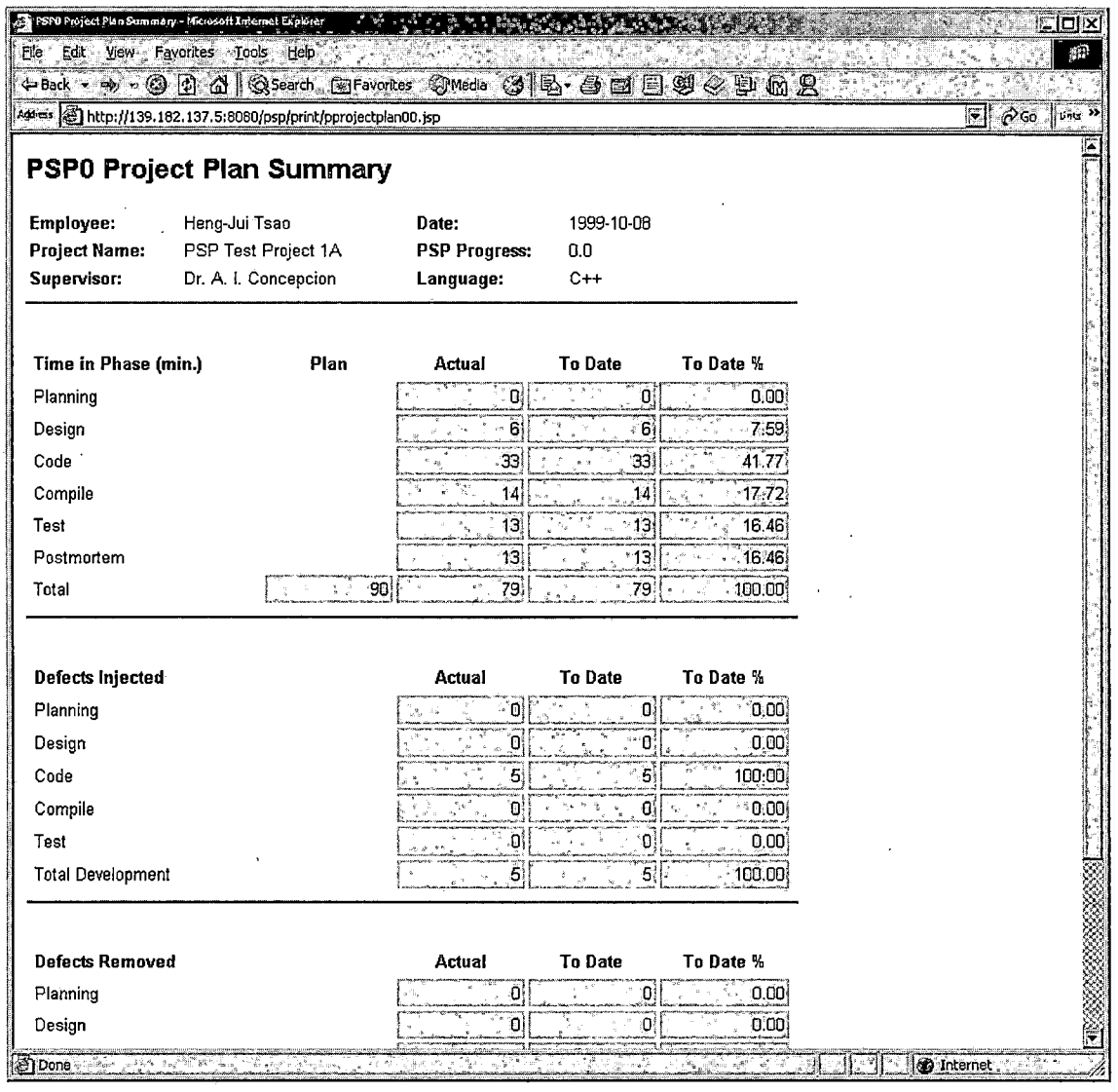


Figure 8. Screen Shot of PSP0 Project Planning Summary



PSP0 Project Plan Summary - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.182.137.5:8080/psp/print/pprojectplan00.jsp

PSP0 Project Plan Summary

Employee:	Heng-Jui Tsao	**Date:**	1999-10-08
Project Name:	PSP Test Project 1A	**PSP Progress:**	0.0
Supervisor:	Dr. A. I. Concepcion	**Language:**	C++

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning		0	0	0.00
Design		6	6	7.59
Code		33	33	41.77
Compile		14	14	17.72
Test		13	13	16.46
Postmortem		13	13	16.46
Total	90	79	79	100.00

Defects Injected	**Actual**	**To Date**	**To Date %**
Planning	0	0	0.00
Design	0	0	0.00
Code	5	5	100.00
Compile	0	0	0.00
Test	0	0	0.00
Total Development	5	5	100.00

Defects Removed	**Actual**	**To Date**	**To Date %**
Planning	0	0	0.00
Design	0	0	0.00

Done Internet

Figure 9. Screen Shot of Print PSP0 Project Planning Summary

<u>1.3.2.4.3 Kit 1 Time Recording Log</u>. The Time Recording Log also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the Time Recording Log. The user can enter data on this PSP applet. The

applet will perform the calculate function (e.g. delta time) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

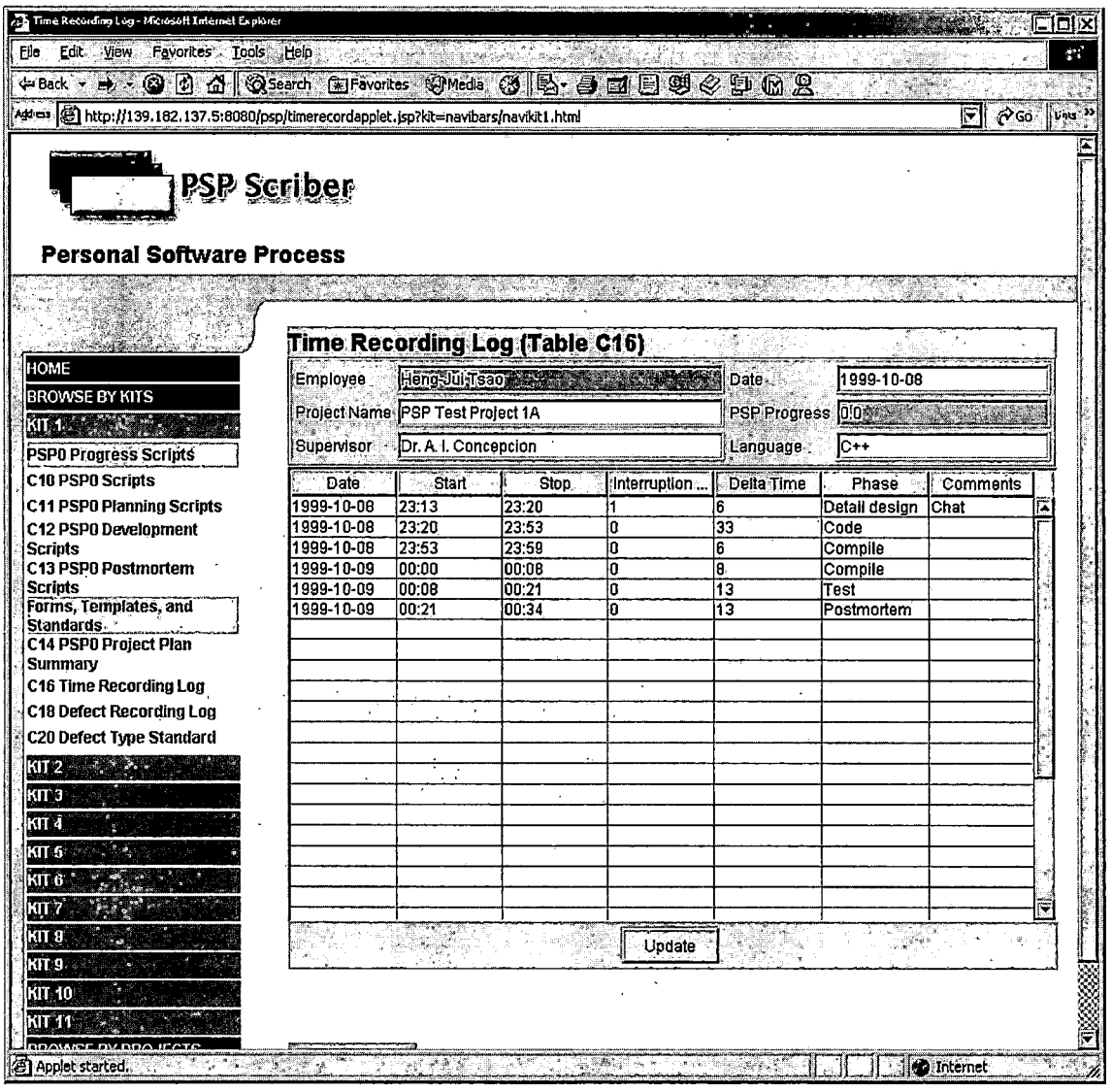
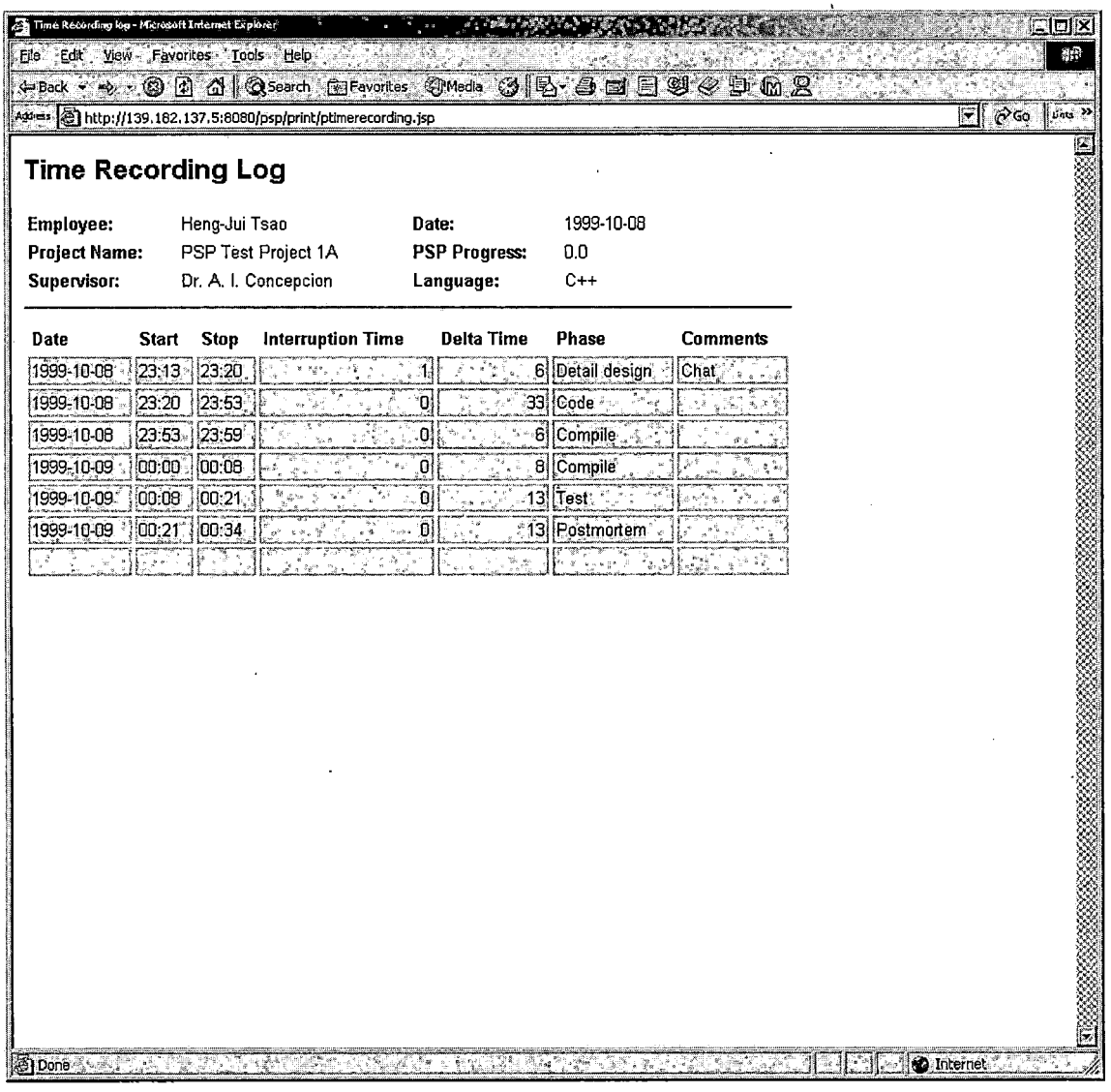


Figure 10. Screen Shot of Time Recording Log



Time Recording log - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.162.137.5:8080/psp/print/ptimerecording.jsp

Time Recording Log

Employee:	Heng-Jui Tsao	**Date:**	1999-10-08
Project Name:	PSP Test Project 1A	**PSP Progress:**	0.0
Supervisor:	Dr. A. I. Concepcion	**Language:**	C++

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
1999-10-08	23:13	23:20	1	6	Detail design	Chat
1999-10-08	23:20	23:53	0	33	Code	
1999-10-08	23:53	23:59	0	6	Compile	
1999-10-09	00:00	00:08	0	8	Compile	
1999-10-09	00:08	00:21	0	13	Test	
1999-10-09	00:21	00:34	0	13	Postmortem	

Done Internet

Figure 11. Screen Shot of Print Time Recording Log

<u>1.3.2.4.4 Kit 1 Defect Recording Log</u>. The Defect Recording Log also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the Defect Recording Log. The user can enter data on this PSP applet. The

applet will perform the calculate function (e.g. defect type) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

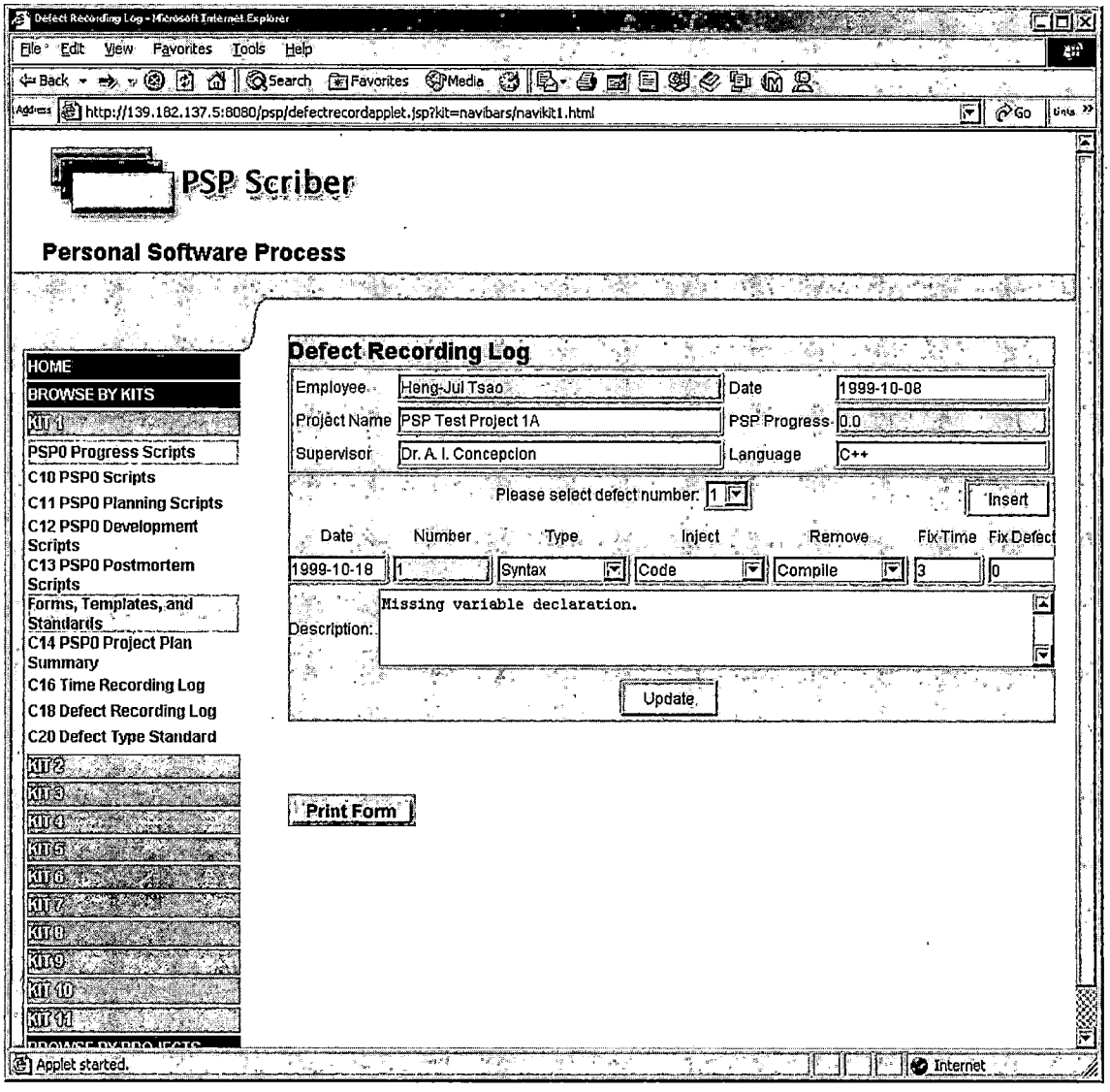
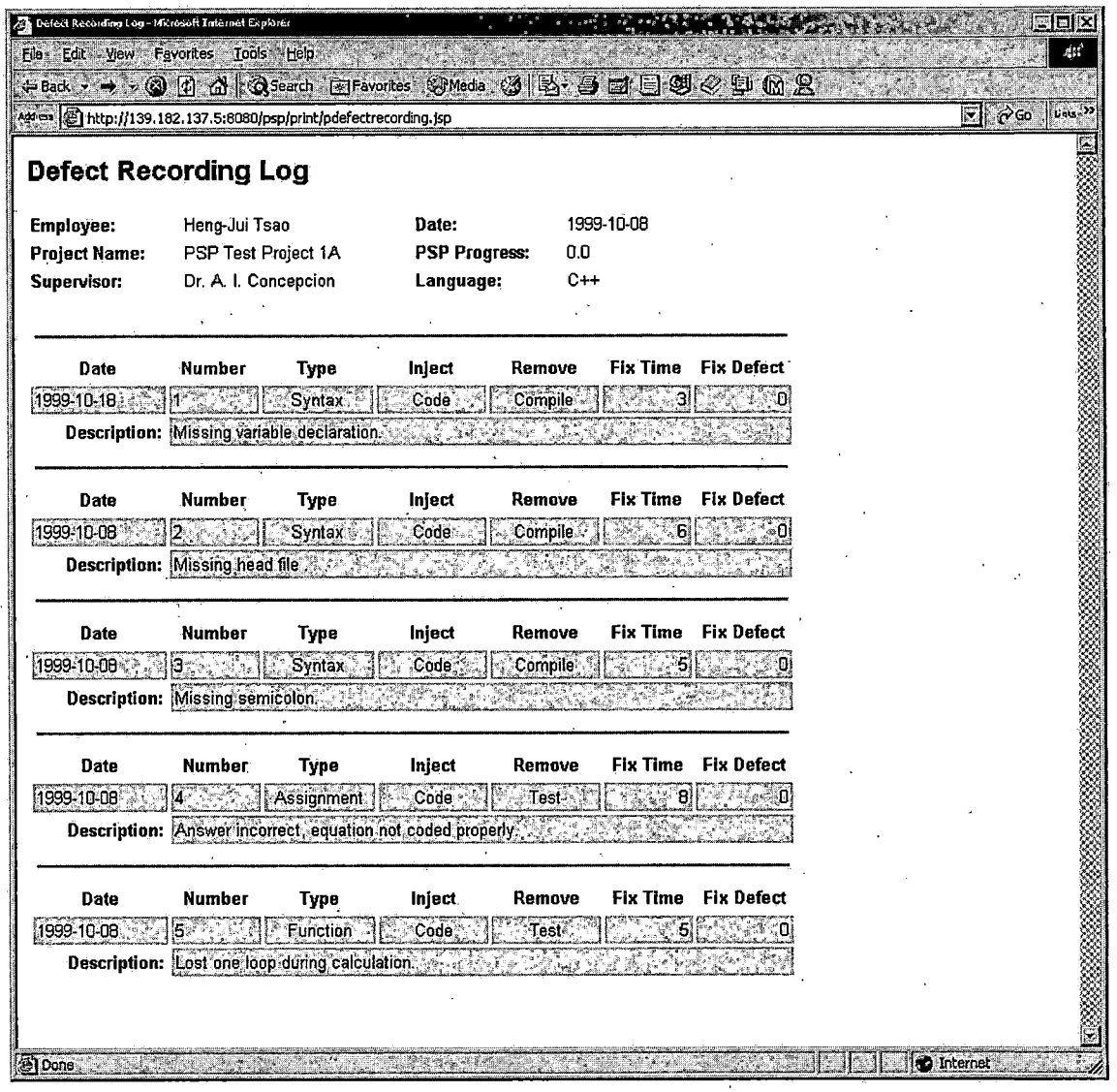


Figure 12. Screen Shot of Defect Recording Log



Defect Recording Log

Employee:	Heng-Jui Tsao	**Date:**	1999-10-08
Project Name:	PSP Test Project 1A	**PSP Progress:**	0.0
Supervisor:	Dr. A. I. Concepcion	**Language:**	C++

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
1999-10-18	1	Syntax	Code	Compile	3	0
Description:	Missing variable declaration.					
1999-10-08	2	Syntax	Code	Compile	6	0
Description:	Missing head file					
1999-10-08	3	Syntax	Code	Compile	5	0
Description:	Missing semicolon.					
1999-10-08	4	Assignment	Code	Test	8	0
Description:	Answer incorrect, equation not coded properly.					
1999-10-08	5	Function	Code	Test	5	0
Description:	Lost one loop during calculation.					

Figure 13. Screen Shot of Print Defect Recording Log

1.3.2.5 Kit 2.

1.3.2.5.1 Kit 2 Main Page. The Kit 2 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and

standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

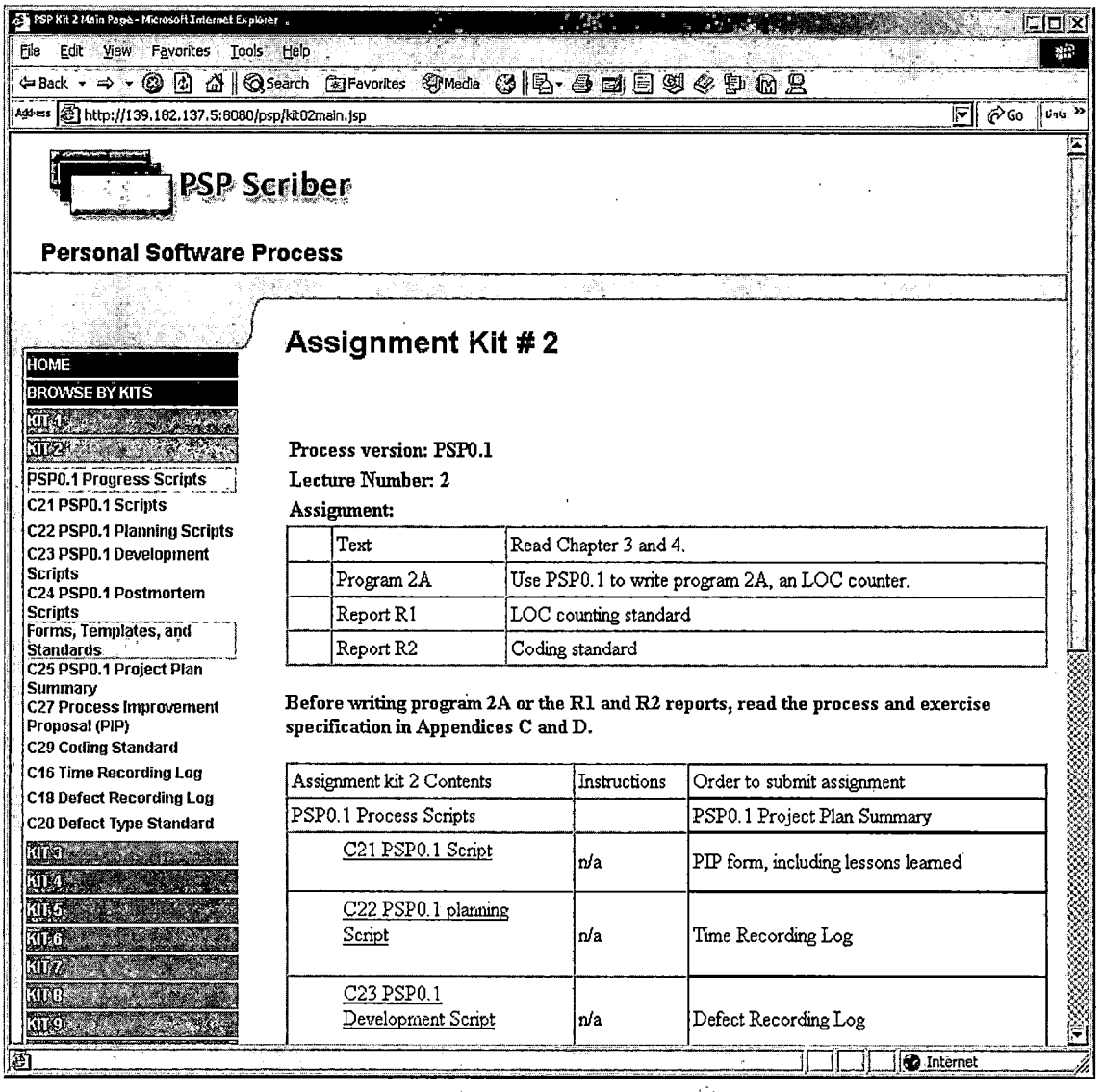


Figure 14. Screen Shot of Kit 2 Main Page

The differences between the kit 2 and the kit 1 are the kit 2 uses the PSP0.1 Project Plan Summary instead of using the PSP0 Project Plan Summary. Also, the kit 2 adds a new forms, the Process Improvement Proposal.

1.3.2.5.2 Kit 2 PSP0.1 Project Plan Summary. The PSP0.1 Project Plan Summary also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the PSP0.1 Project Plan Summary. The user can enter data on this PSP applet. The applet will perform the calculate function (e.g. to date planning time) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

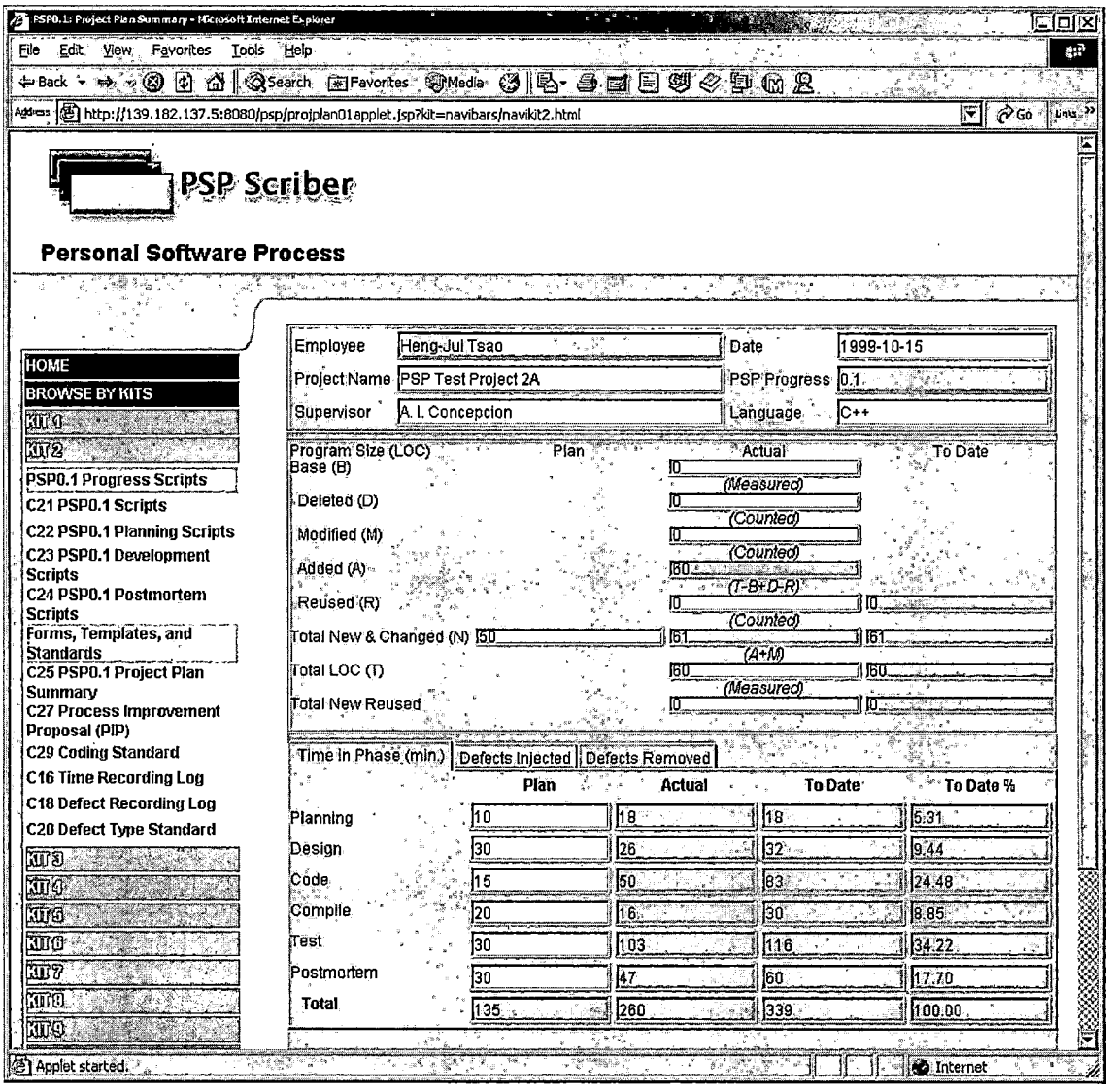


Figure 15. Screen Shot of Kit 2 PSP0.1 Project Planning Summary

Cannot find server - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.182.137.5:8080/psp/print/pprojectplan01.jsp Go

PSP0.1 Project Plan Summary

Employee:	Heng-Jui Tsao	**Date:**	1999-10-15
Project Name:	PSP Test Project 2A	**PSP Progress:**	0.1
Supervisor:	A. I. Concepcion	**Language:**	C++

Program size (LOC)	*Plan*	*Actual*	*To Date*
Base (B)		0	
		(Measured)	
Deleted (D)		0	
		(Counted)	
Modified (M)		0	
		(Counted)	
Added (A)		60	
		(T - B + D - R)	
Reused		0	0
		(Counted)	
Total New & Changed (N)	50	61	61
		(A + M)	
Total LOC (T)		60	60
		(Measured)	
Total New Reused		0	0

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning	10	18	18	5.31
Design	30	26	32	9.44
Code	15	50	83	24.48
Compile	20	16	30	8.85

Done Internet

Figure 16. Screen Shot of Print PSP0.1 Project Planning Summary

1.3.2.5.3 Kit 2 Process Improvement Proposal.

The Process Improvement Proposal also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the Process

Improvement Proposal. The user can enter data on this PSP applet. The applet will perform the calculate function (e.g. problem description) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

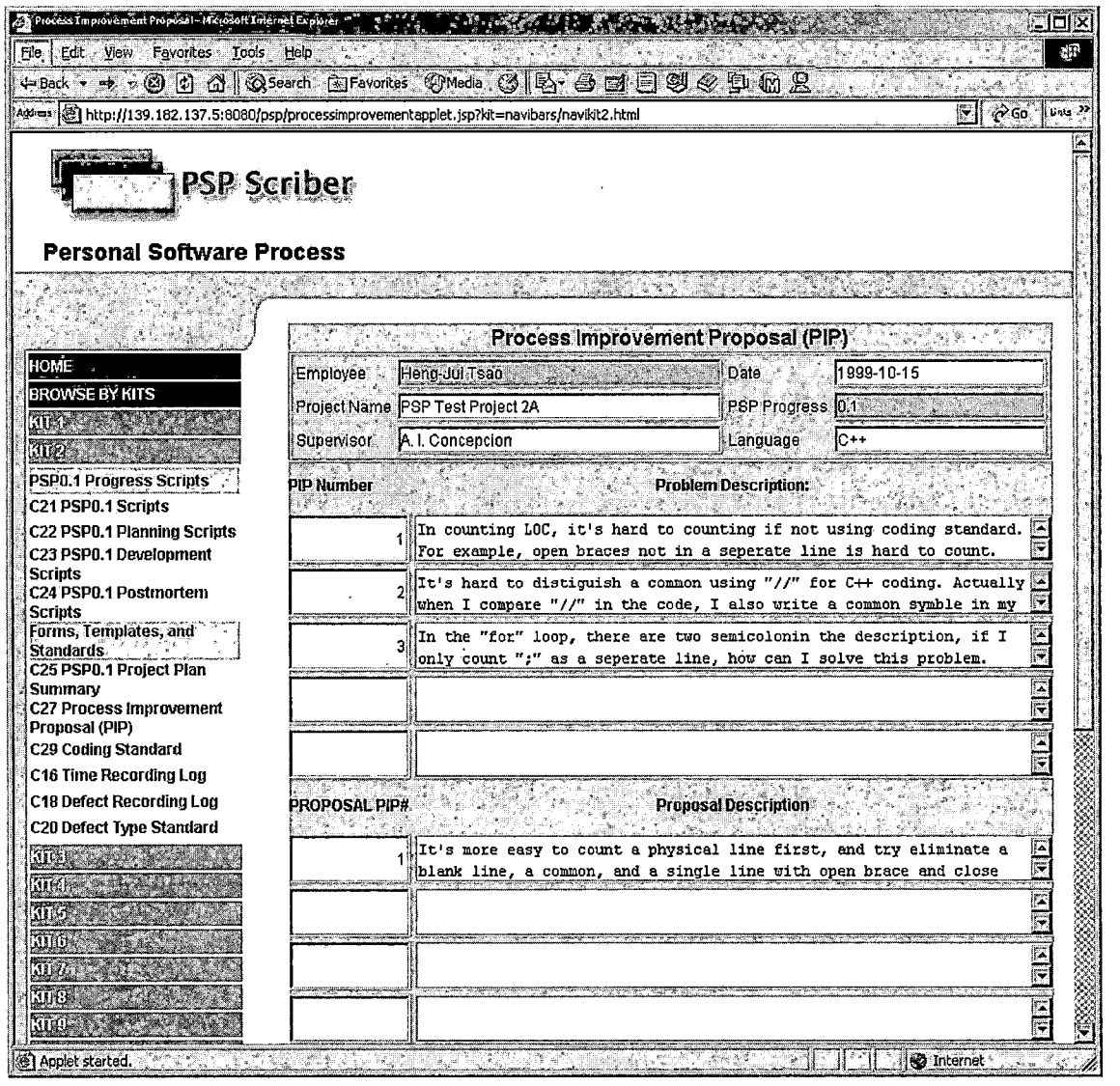
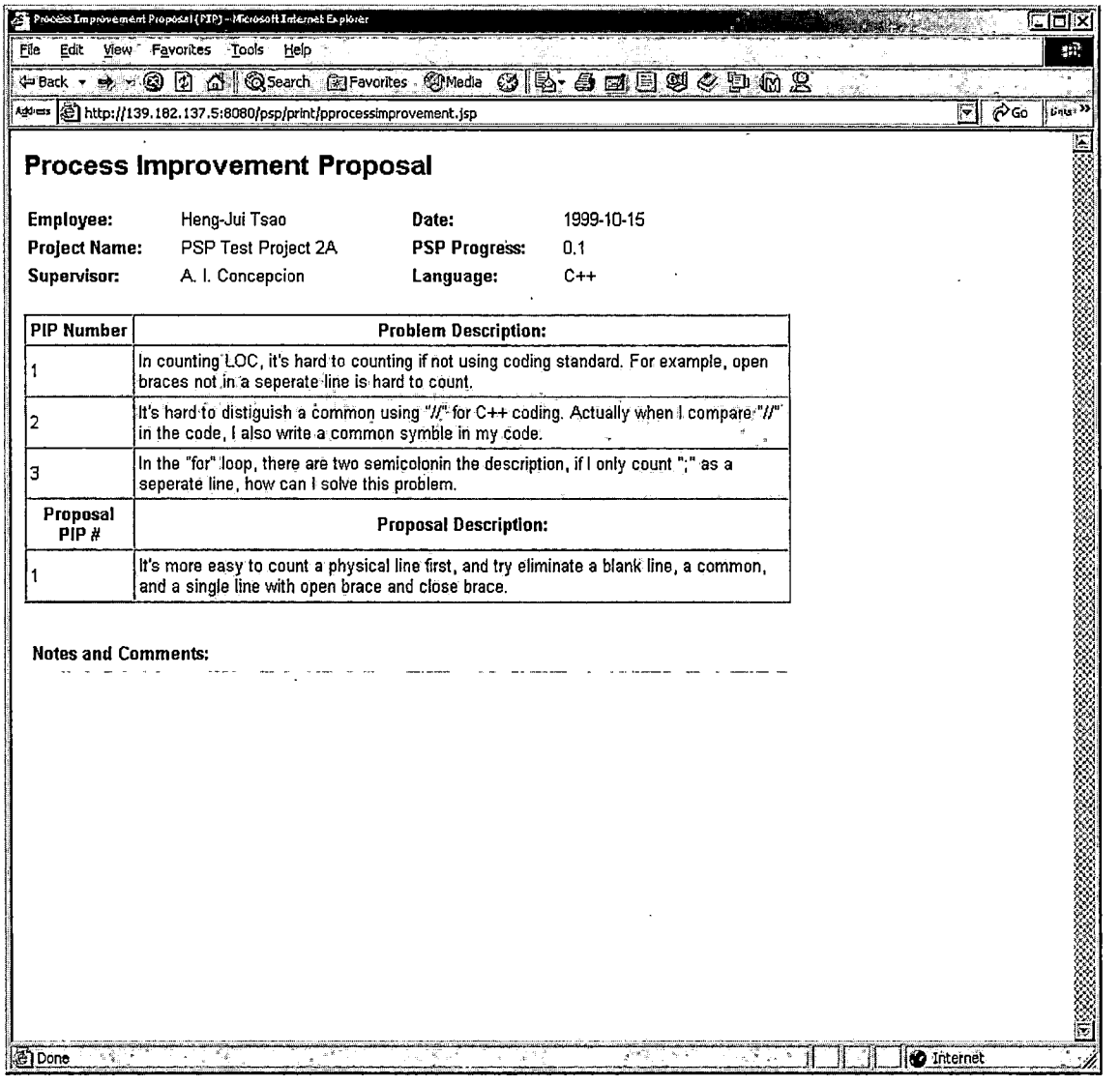


Figure 17. Screen Shot of Process Improvement Proposal



Process Improvement Proposal (PIP) - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.182.137.5:8080/psp/print/pprocessimprovement.jsp

Process Improvement Proposal

Employee:	Heng-Jui Tsao	**Date:**	1999-10-15
Project Name:	PSP Test Project 2A	**PSP Progress:**	0.1
Supervisor:	A. I. Concepcion	**Language:**	C++

PIP Number	Problem Description:
1	In counting LOC, it's hard to counting if not using coding standard. For example, open braces not in a seperate line is hard to count.
2	It's hard to distiguish a common using "//" for C++ coding. Actually when I compare "//" in the code, I also write a common symble in my code.
3	In the "for" loop, there are two semicolonin the description, if I only count ";" as a seperate line, how can I solve this problem.
Proposal PIP #	**Proposal Description:**
1	It's more easy to count a physical line first, and try eliminate a blank line, a common, and a single line with open brace and close brace.

Notes and Comments:

Done Internet

Figure 18. Screen Shot of Print Process Improvement Proposal

1.3.2.5.4 Kit 2 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.5.5 Kit 2 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Time Recording Log.

1.3.2.6 Kit 3.

1.3.2.6.1 Kit 3 Main Page. The Kit 3 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

The kit 3 has all the forms and templates as same as the kit 2.

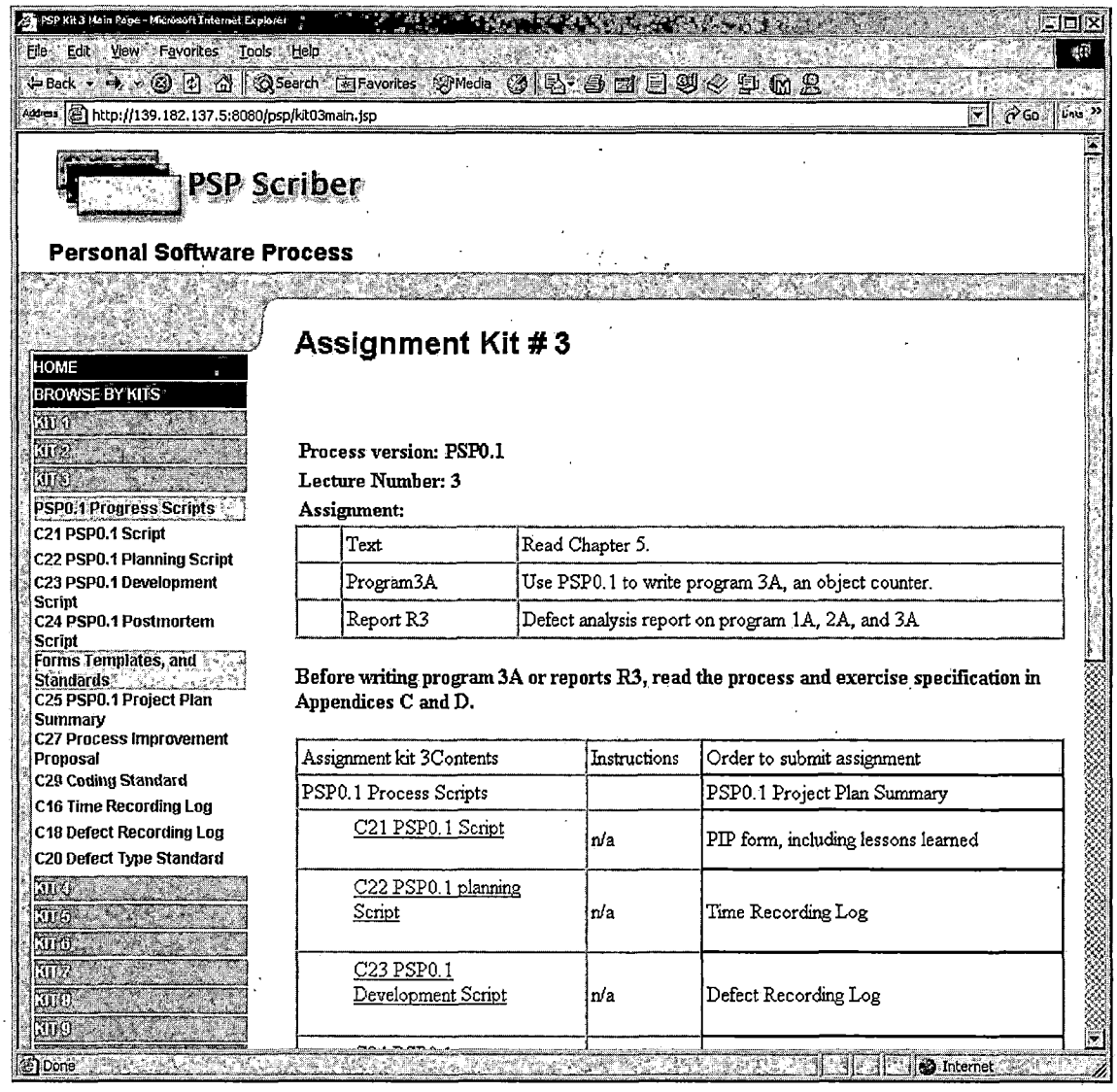


Figure 19. Screen Shot of Kit 3 Main Page

1.3.2.6.2 Kit 3 PSP0.1 Project Plan Summary. As shown on 1.3.2.5.2 Kit 2 PSP0.1 Project Plan Summary.

1.3.2.6.3 Kit 3 Process Improvement Proposal. As shown on 1.3.2.5.3 Kit 2 Process Improvement Proposal.

1.3.2.6.4 Kit 3 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.6.5 Kit 3 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.2.7 Kit 4.

1.3.2.7.1 Kit 4 Main Page. The Kit 4 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

The differences between the kit 4 and the kit 3 are the kit 4 uses the PSP1 Project Plan Summary instead of using the PSP0.1 Project Plan Summary. Also, the kit 4 adds two new templates, the Test Report Template and the Size Estimating Template.

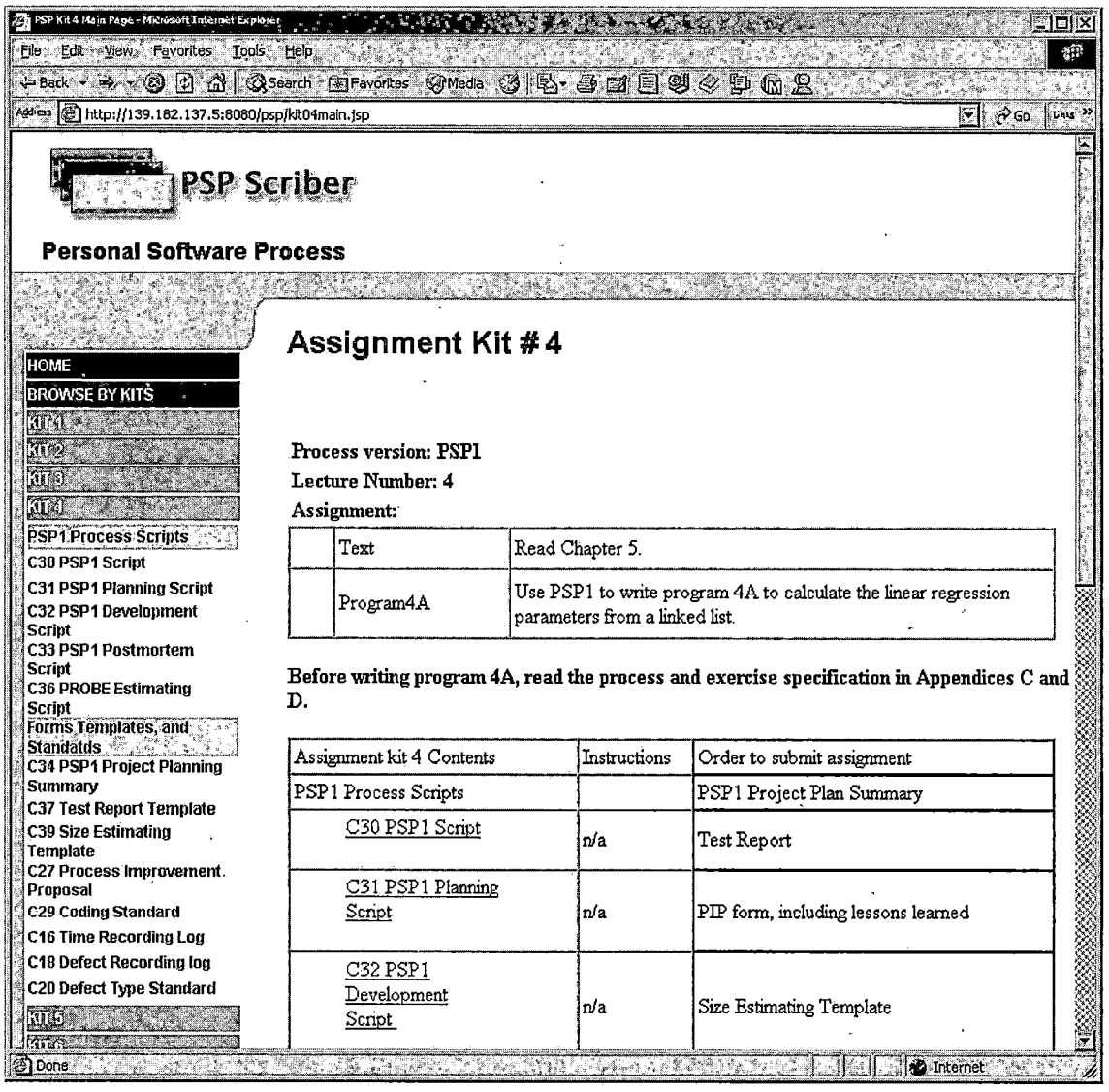


Figure 20. Screen Shot of Kit 4 Main Page

1.3.2.7.2 Kit 4 PSP1 Project Plan Summary. The PSP1 Project Plan Summary also has two frames on it. The left frame provides the same function as Main Page. The Left frame provides the PSP applet to show the PSP1 Project Plan Summary. The user can enter data on this PSP

applet. The applet will perform the calculate function (e.g. to date planning time) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

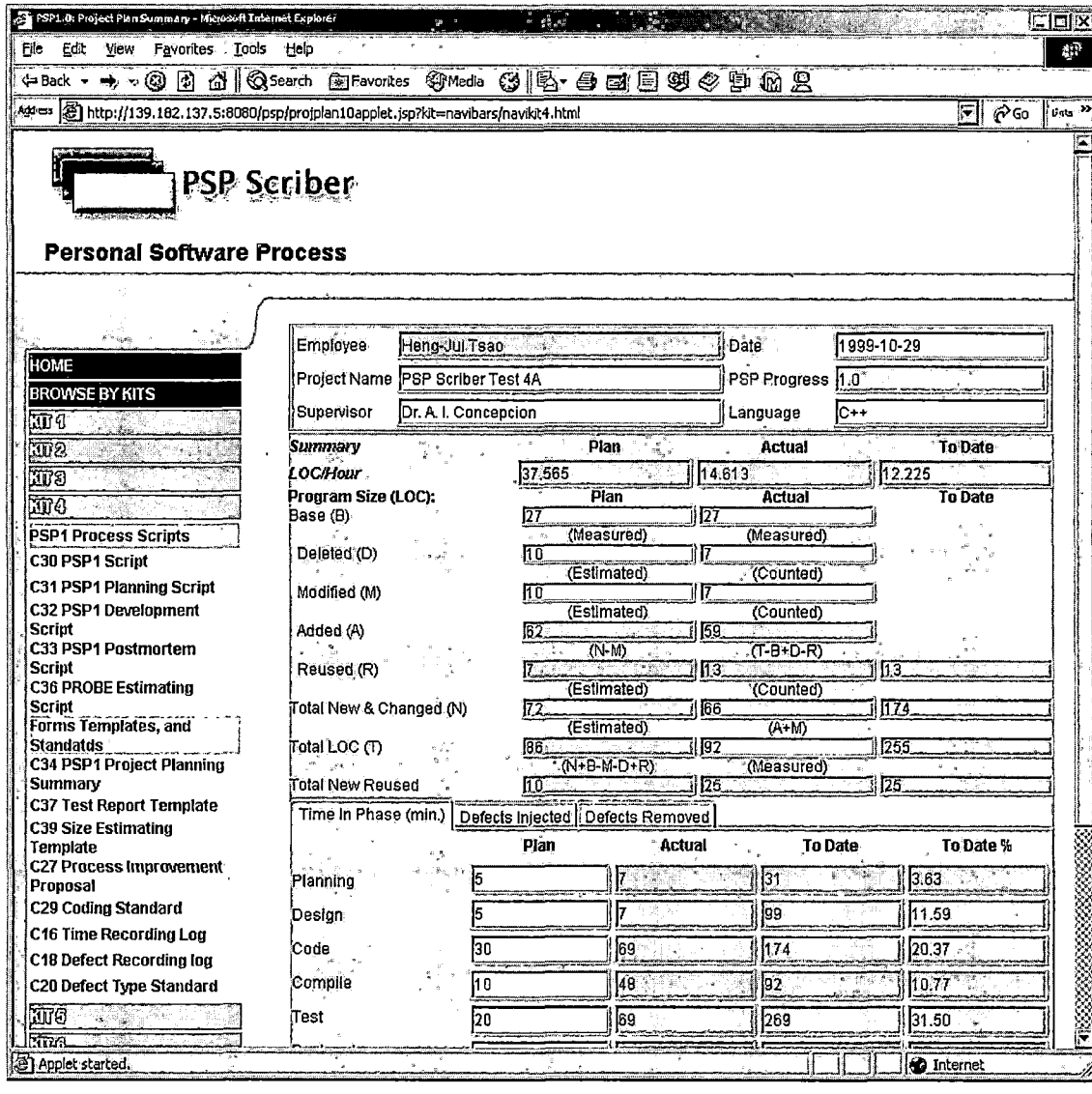
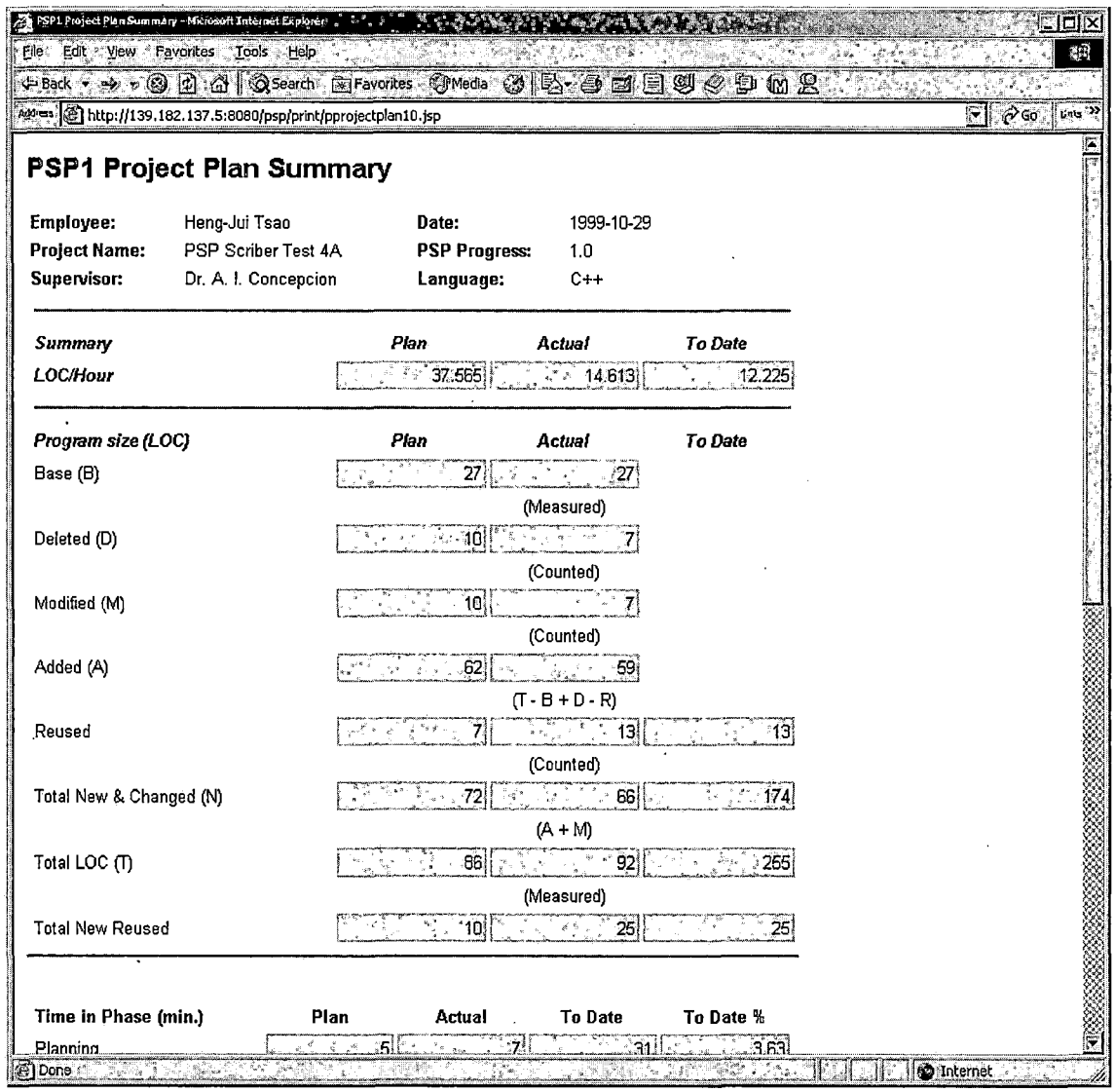


Figure 21. Screen Shot of PSP1 Project Planning Summary



PSP1 Project Plan Summary - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.182.137.5:8080/psp/print/pprojectplan10.jsp

PSP1 Project Plan Summary

Employee:	Heng-Jui Tsao	**Date:**	1999-10-29
Project Name:	PSP Scriber Test 4A	**PSP Progress:**	1.0
Supervisor:	Dr. A. I. Concepcion	**Language:**	C++

Summary	*Plan*	*Actual*	*To Date*
LOC/Hour	37.565	14.613	12.225

Program size (LOC)	*Plan*	*Actual*	*To Date*
Base (B)	27	27	
		(Measured)	
Deleted (D)	10	7	
		(Counted)	
Modified (M)	10	7	
		(Counted)	
Added (A)	62	59	
		(T - B + D - R)	
Reused	7	13	13
		(Counted)	
Total New & Changed (N)	72	66	174
		(A + M)	
Total LOC (T)	86	92	255
		(Measured)	
Total New Reused	10	25	25

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning	5	7	31	3.63

Done Internet

Figure 22. Screen Shot of Print PSP1 Project Planning Summary

1.3.2.7.3 Kit 4 Test Report Template. The Test Report Template also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the Test Report Template. The user can enter data on this PSP applet. The

applet will perform the calculate function (e.g. test condition) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

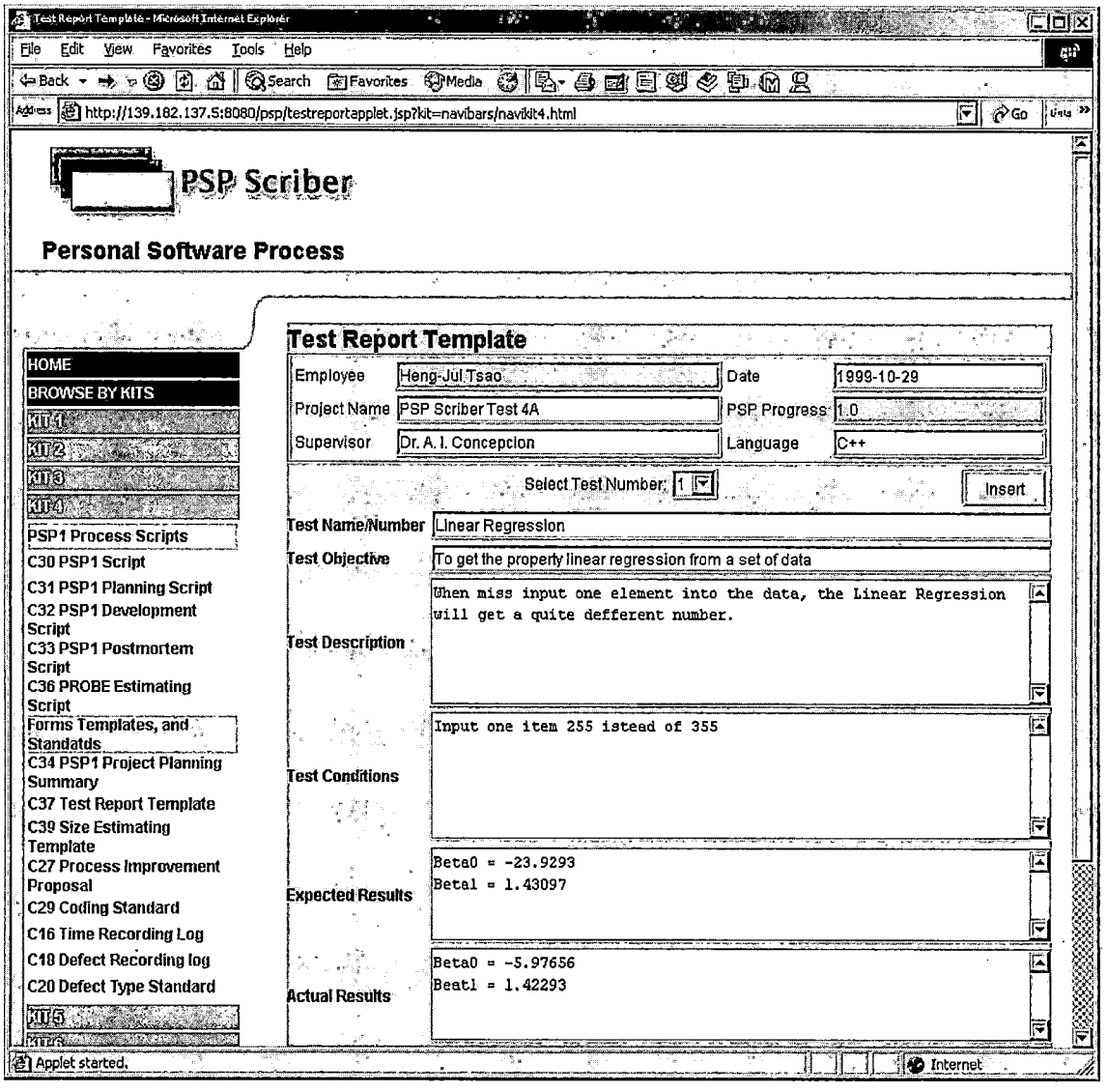
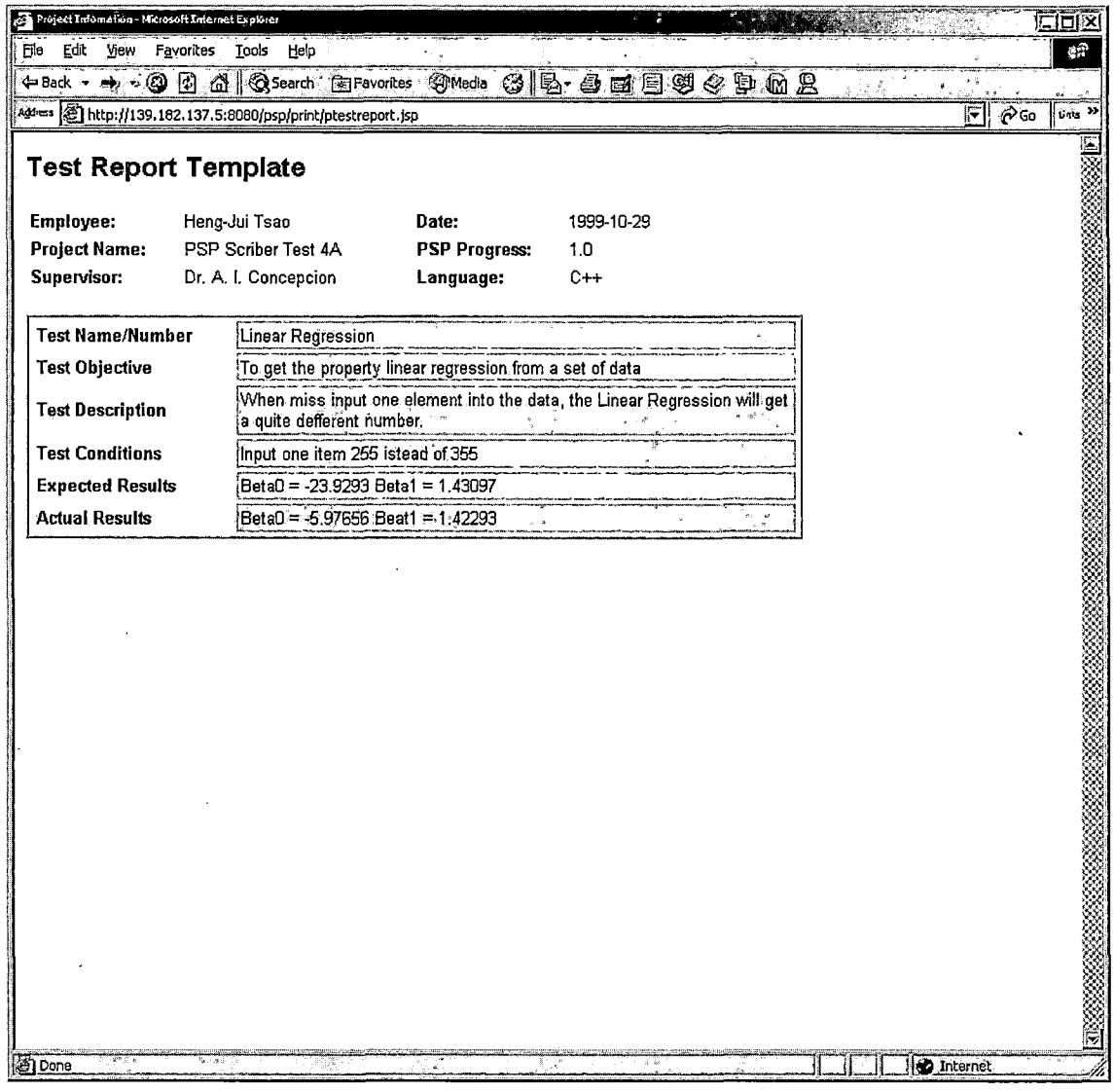


Figure 23. Screen Shot of Test Report Template



Project Information - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.182.137.5:8080/psp/print/ptestreport.jsp

Go Links

Test Report Template

Employee:	Heng-Jui Tsao	**Date:**	1999-10-29
Project Name:	PSP Scriber Test 4A	**PSP Progress:**	1.0
Supervisor:	Dr. A. I. Concepcion	**Language:**	C++

Test Name/Number	Linear Regression
Test Objective	To get the property linear regression from a set of data
Test Description	When miss input one element into the data, the Linear Regression will get a quite defferent number.
Test Conditions	Input one item 255 istead of 355
Expected Results	Beta0 = -23.9293 Beta1 = 1.43097
Actual Results	Beta0 = -5.97656 Beat1 = 1.42293

Done Internet

Figure 24. Screen Shot of Print Test Report Template

1.3.2.7.4 Kit 4 Size Estimating Template. The Size Estimating Template also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the Size Estimating Template. The user can enter data on this PSP

applet. The applet will perform the calculate function (e.g. regression parameter) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

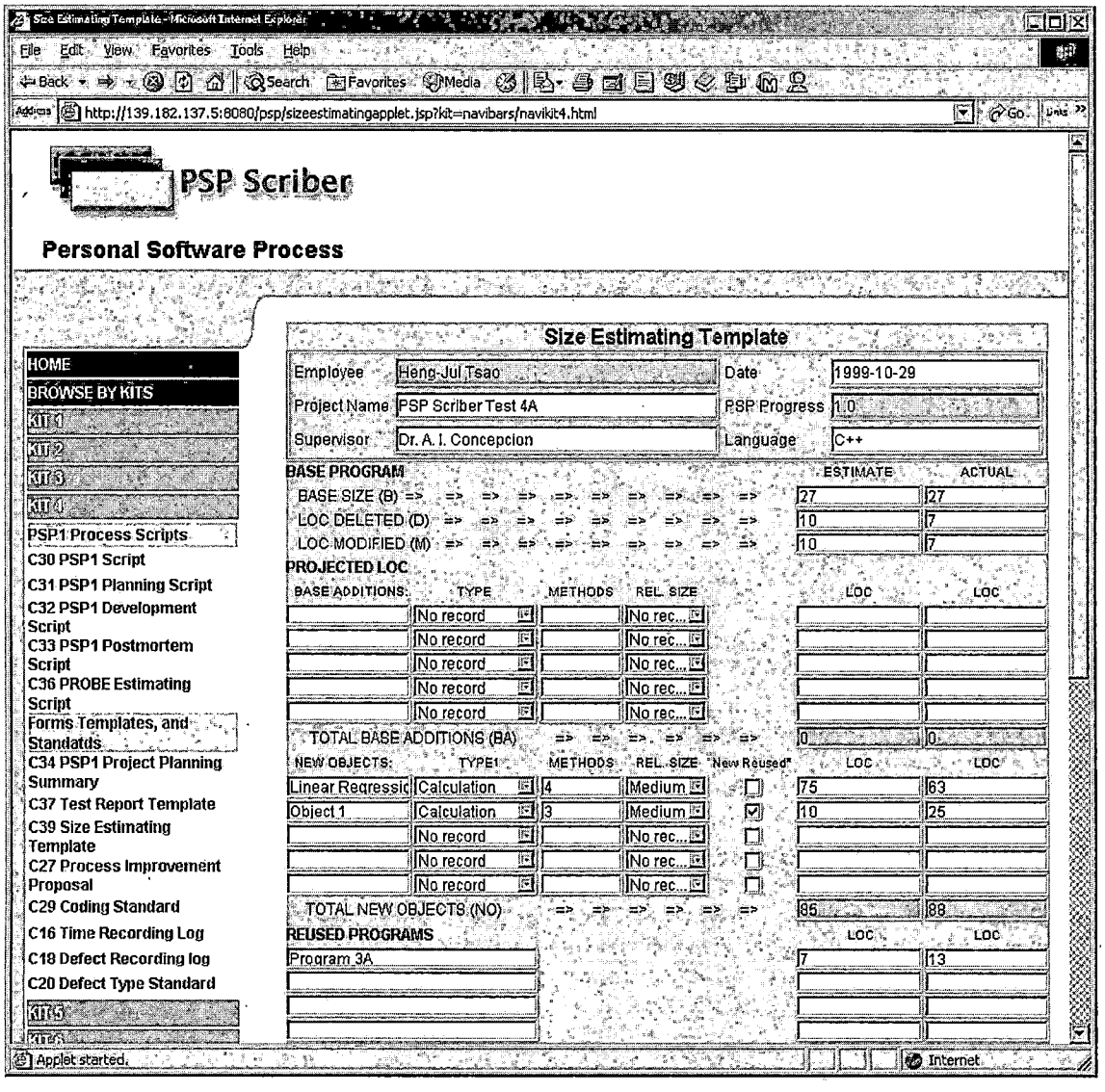
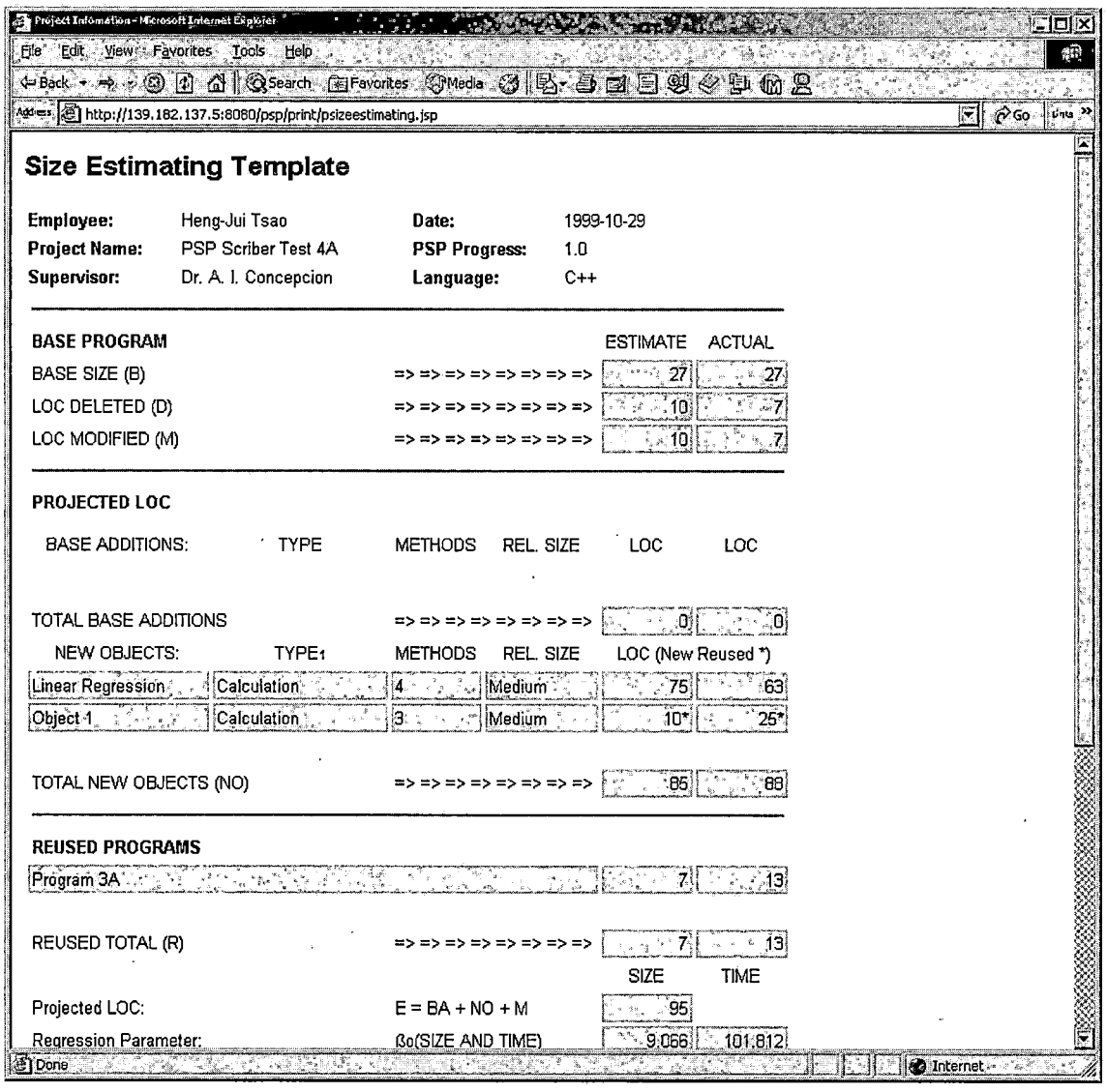


Figure 25. Screen Shot of Size Estimating Template



Project Information - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.182.137.5:8080/psp/print/psizeestimating.jsp

Size Estimating Template

Employee:	Heng-Jui Tsao	**Date:**	1999-10-29
Project Name:	PSP Scriber Test 4A	**PSP Progress:**	1.0
Supervisor:	Dr. A. I. Concepcion	**Language:**	C++

BASE PROGRAM		ESTIMATE	ACTUAL
BASE SIZE (B)	=> => => => => => => =>	27	27
LOC DELETED (D)	=> => => => => => => =>	10	7
LOC MODIFIED (M)	=> => => => => => => =>	10	7

PROJECTED LOC

BASE ADDITIONS:	TYPE	METHODS	REL. SIZE	LOC	LOC
TOTAL BASE ADDITIONS		=> => => => => => => =>		0	0
NEW OBJECTS:	TYPE1	METHODS	REL. SIZE	LOC (New Reused *)	
Linear Regression	Calculation	4	Medium	75	63
Object 1	Calculation	3	Medium	10*	25*
TOTAL NEW OBJECTS (NO)		=> => => => => => => =>		85	88

REUSED PROGRAMS

Program 3A		7	13
REUSED TOTAL (R)	=> => => => => => => =>	7	13
		SIZE	TIME
Projected LOC:	E = BA + NO + M	95	
Regression Parameter:	ßo(SIZE AND TIME)	9.066	101.812

Done Internet

Figure 26. Screen Shot of Print Size Estimate Template

1.3.2.7.5 Kit 4 Process Improvement Proposal. As shown on 1.3.2.5.3 Kit 2 Process Improvement Proposal.

1.3.2.7.6 Kit 4 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.7.7 Kit 4 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.2.8 Kit 5.

1.3.2.8.1 Kit 5 Main Page. The Kit 5 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

The differences between the kit 5 and the kit 4 are the kit 5 uses the PSP1.1 Project Plan Summary instead of using the PSP1 Project Plan Summary. Also, the kit 5 adds two new templates, the Task Planning Template and the Schedule Planning Template.

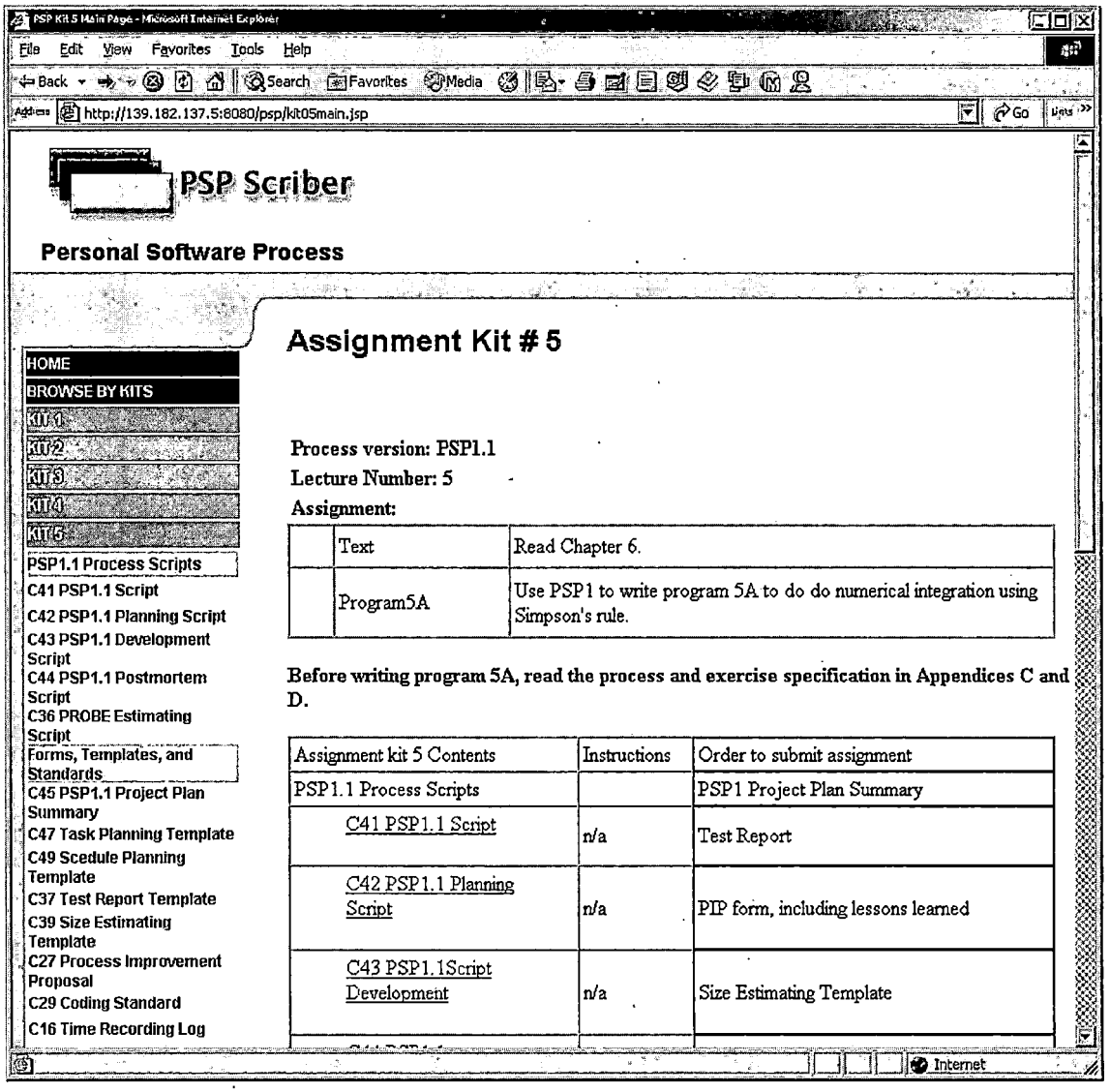


Figure 27. Screen Shot of Kit 5 Main Page

1.3.2.8.2 Kit 5 PSP1.1 Project Plan Summary. The PSP1 Project Plan Summary also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the PSP1.1 Project Plan Summary. The user can enter data on this PSP

applet. The applet will perform the calculate function (e.g. to date planning time) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

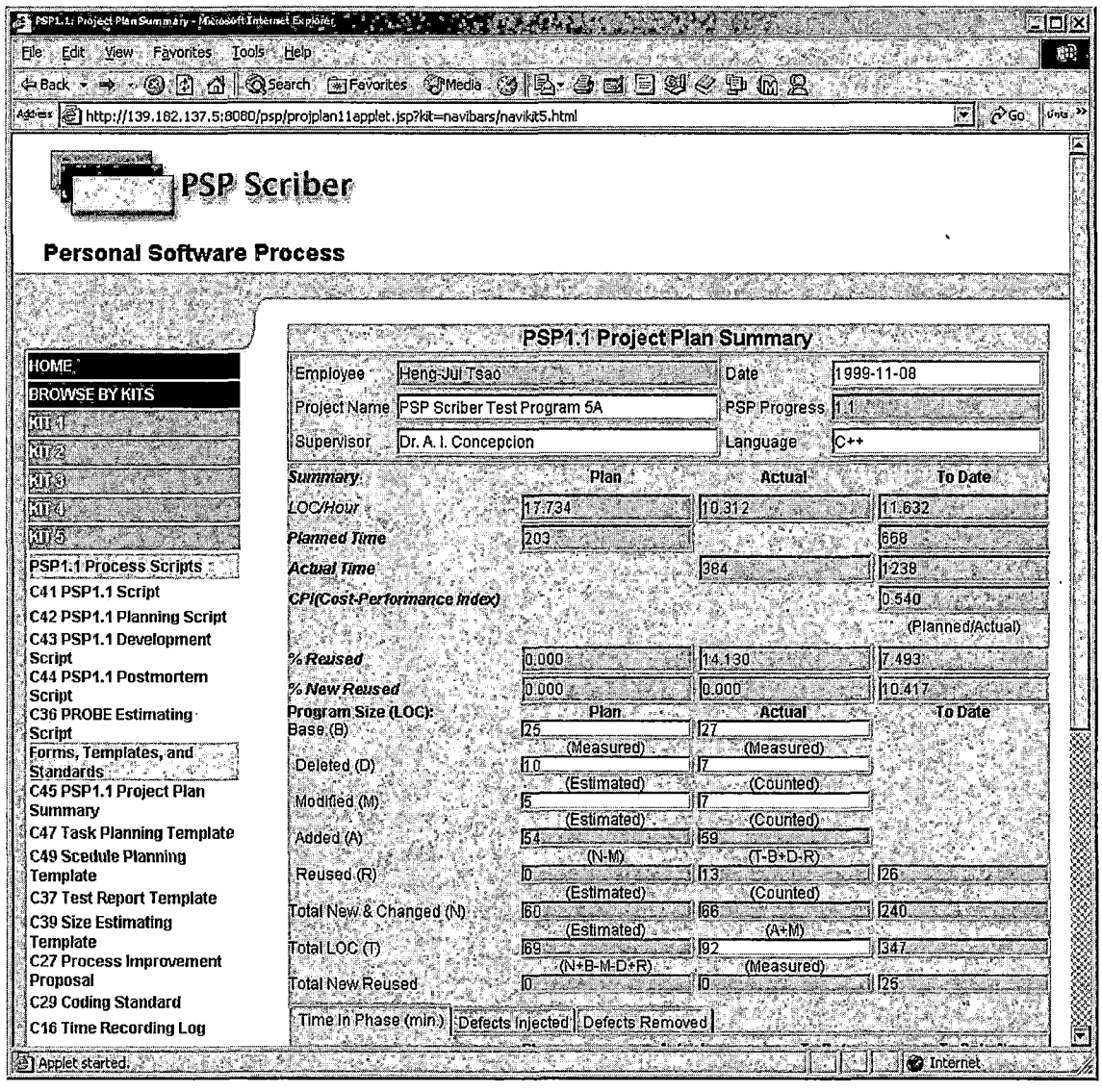


Figure 28. Screen Shot of Kit 5 PSP1.1 Project Planning Summary

PSP1.1 Project Plan Summary - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address http://139.182.137.5:8080/psp/print/pprojectplan11.jsp

PSP1.1 Project Plan Summary

Employee:	Heng-Jui Tsao	**Date:**	1999-11-08
Project Name:	PSP Scriber Test Program 5A	**PSP Progress:**	1.1
Supervisor:	Dr. A. I. Concepcion	**Language:**	C++

Summary	**Plan**	**Actual**	**To Date**
LOC/Hour	17.734	10.312	11.632
Planned Time	203		668
Actual Time		384	1238
CPI(Cost-Performance Index)			0.540
			(Planned/Actual)
% Reused	0.000	14.130	7.493
% New Reused	0.000	0.000	10.417

Program size (LOC)	***Plan***	***Actual***	***To Date***
Base (B)	25	27	
		(Measured)	
Deleted (D)	10	7	
		(Counted)	
Modified (M)	5	7	
		(Counted)	
Added (A)	54	59	
		(T - B + D - R)	
Reused	0	13	26
		(Counted)	
Total New & Changed (N)	60	66	240
		(A + M)	

Done Internet

Figure 29. Screen Shot of Print PSP1.1 Project Planning Summary

1.3.2.8.3 Kit 5 Task Plan Template. The Task Plan Template also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the Task Plan Template. The user can enter data on this PSP applet. The

applet will perform the calculate function (e.g. cumulative earned value) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

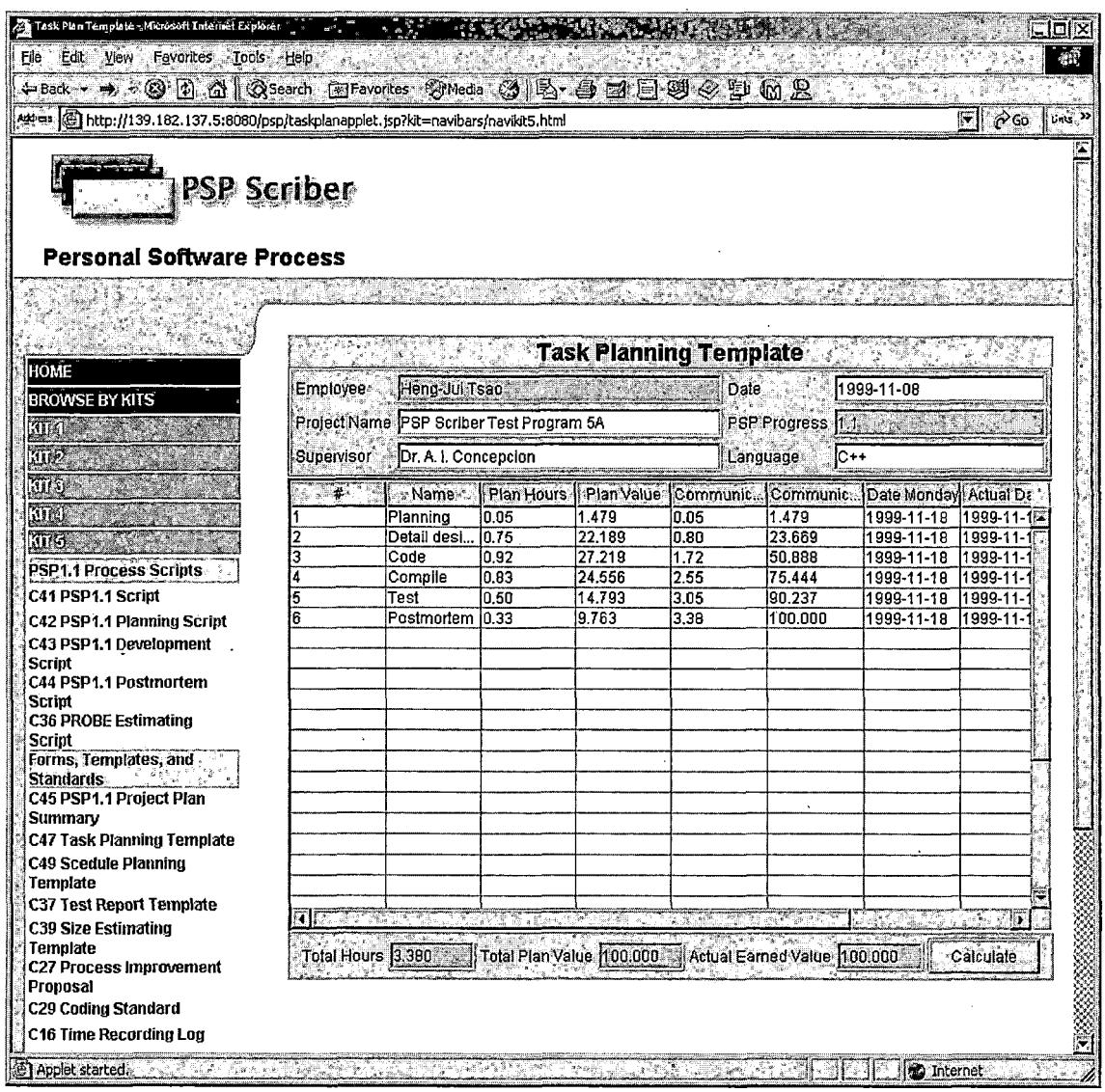
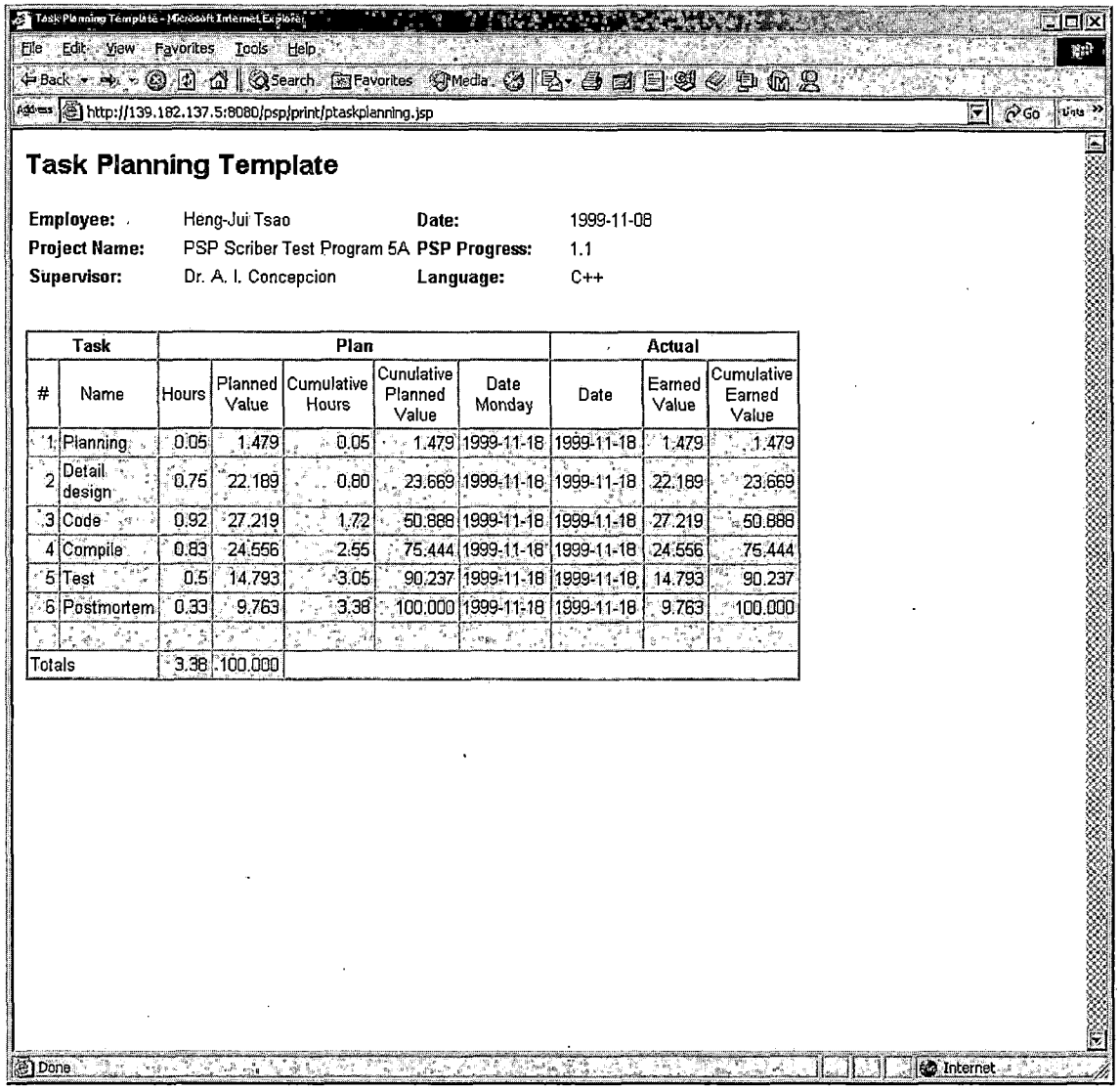


Figure 30. Screen Shot of Task Plan Template



Task Planning Template

Employee: Heng-Jui Tsao **Date:** 1999-11-08
Project Name: PSP Scriber Test Program 5A **PSP Progress:** 1.1
Supervisor: Dr. A. I. Concepcion **Language:** C++

Task		Plan					Actual		
#	Name	Hours	Planned Value	Cumulative Hours	Cunulative Planned Value	Date Monday	Date	Earned Value	Cumulative Earned Value
1	Planning	0.05	1.479	0.05	1.479	1999-11-18	1999-11-18	1.479	1.479
2	Detail design	0.75	22.189	0.80	23.669	1999-11-18	1999-11-18	22.189	23.669
3	Code	0.92	27.219	1.72	50.888	1999-11-18	1999-11-18	27.219	50.888
4	Compile	0.83	24.556	2.55	75.444	1999-11-18	1999-11-18	24.556	75.444
5	Test	0.5	14.793	3.05	90.237	1999-11-18	1999-11-18	14.793	90.237
6	Postmortem	0.33	9.763	3.38	100.000	1999-11-18	1999-11-18	9.763	100.000
Totals		3.38	100.000						

Figure 31. Screen Shot of Print Task Plan Template

1.3.2.8.4 Kit 5 Schedule Planning Template. The Schedule Plan Template also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the Schedule Plan Template. User can enter data on this PSP applet. The

applet will perform the calculate function (e.g. cumulative earned value) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

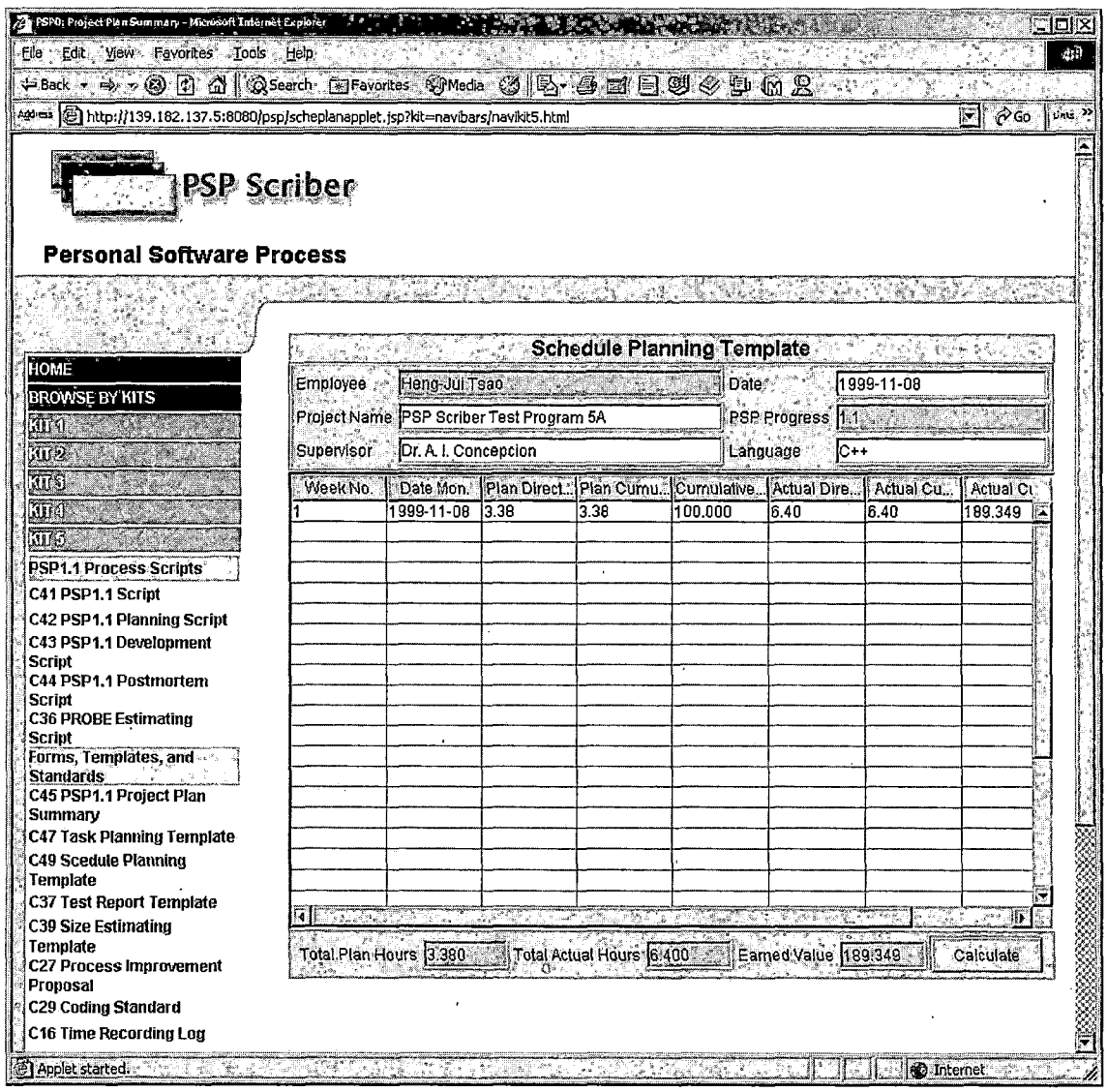
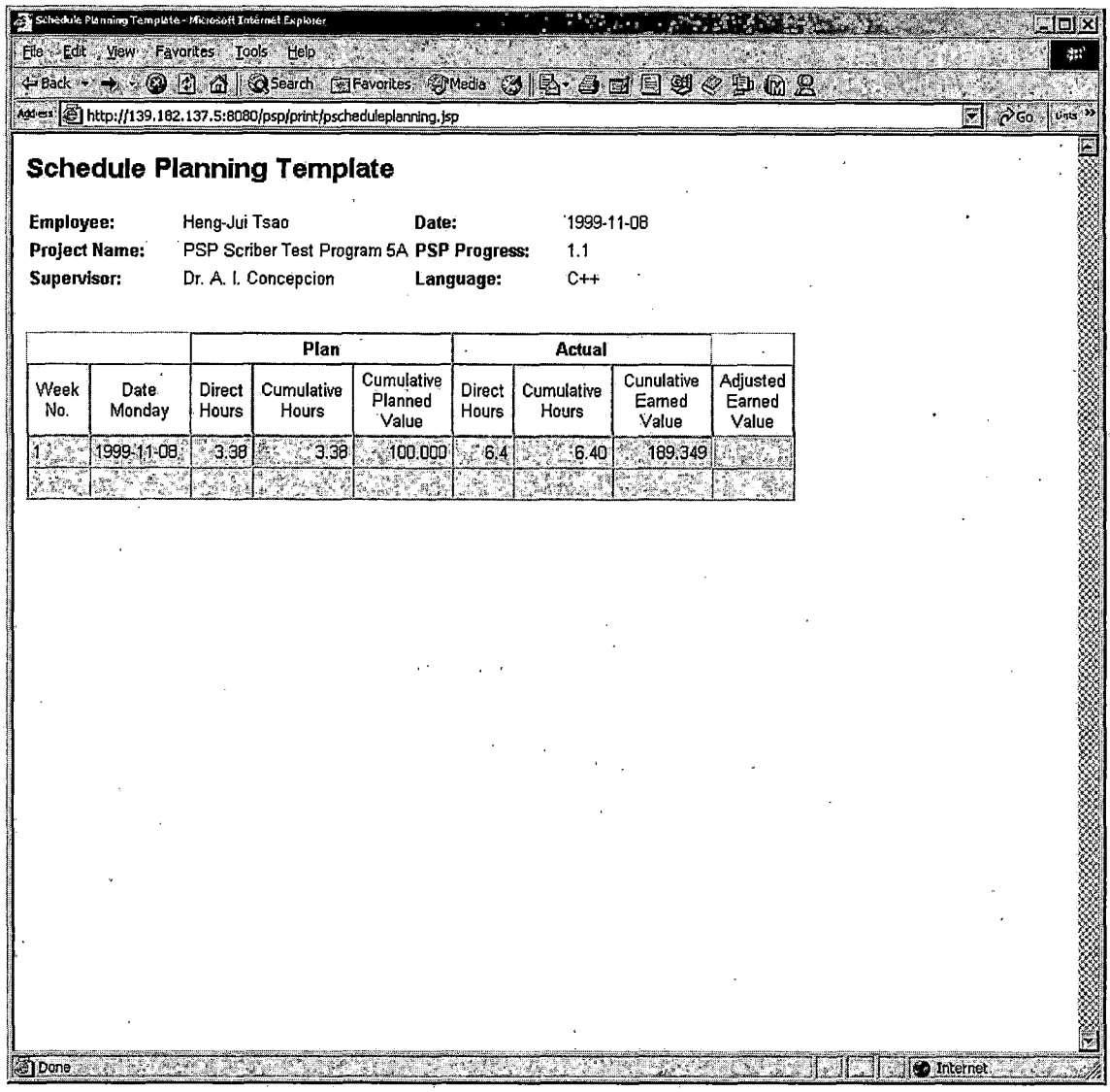


Figure 32. Screen Shot of Schedule Planning Template



Schedule Planning Template

Employee: Heng-Jui Tsao **Date:** 1999-11-08
Project Name: PSP Scriber Test Program 5A **PSP Progress:** 1.1
Supervisor: Dr. A. I. Concepcion **Language:** C++

		Plan			Actual			
Week No.	Date Monday	Direct Hours	Cumulative Hours	Cumulative Planned Value	Direct Hours	Cumulative Hours	Cunulative Earned Value	Adjusted Earned Value
1	1999-11-08	3.38	3.38	100.000	6.4	6.40	189.349	

Figure 33. Screen Shot of Print Schedule Plan Template

1.3.2.8.5 Kit 5 Test Report Template. As shown on 1.3.2.7.3 Kit 4 Test Report Template.

1.3.2.8.6 Kit 5 Size Estimating Template. As shown on 1.3.2.7.4 Kit 4 Size Estimating Template.

1.3.2.8.7 Kit 5 Process Improvement Proposal. As shown on 1.3.2.5.3 Kit 2 Process Improvement Proposal.

1.3.2.8.8 Kit 5 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.8.9 Kit 5 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.2.9 Kit 6.

1.3.2.9.1 Kit 6 Main Page. The Kit 6 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

The kit 6 has all the forms and templates as same as the kit 5.

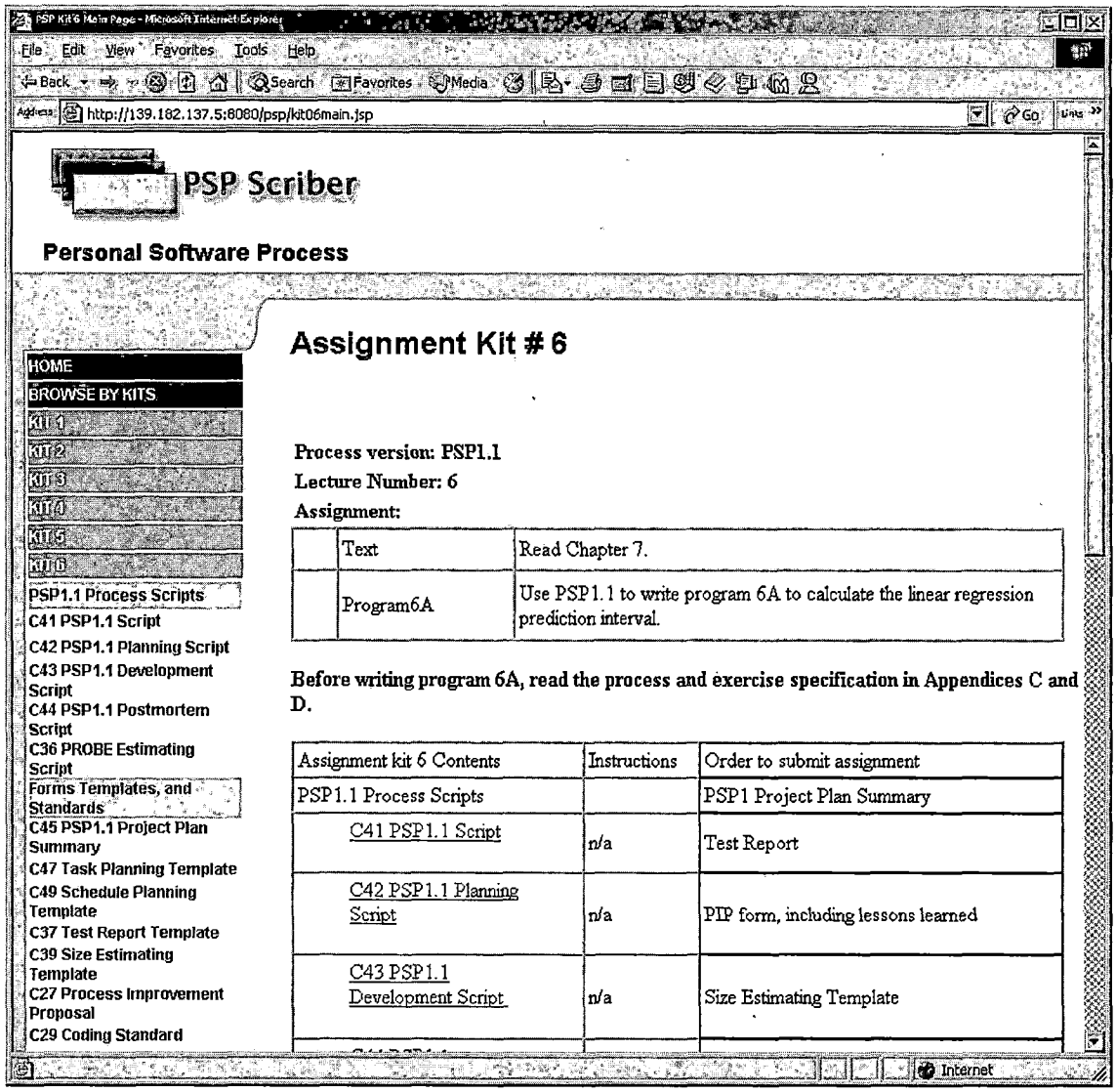


Figure 34. Screen Shot of Kit 6 Main Page

1.3.2.9.2 Kit 6 PSP1.1 Project Plan Summary. As shown on 1.3.2.8.2 Kit 5 PSP1.1 Project Plan Summary.

1.3.2.9.3 Kit 6 Task Planning Template. As shown on 1.3.2.8.3 Kit 5 Task Planning Template.

1.3.2.9.4 Kit 6 Schedule Planning Template. As shown on 1.3.2.8.4 Kit 5 Schedule Planning Template.

1.3.2.9.5 Kit 6 Test Report Template. As shown on 1.3.2.7.3 Kit 4 Test Report Template.

1.3.2.9.6 Kit 6 Size Estimating Template. As shown on 1.3.2.7.4 Kit 4 Size Estimating Template.

1.3.2.9.7 Kit 6 Process Improvement Proposal. As shown on 1.3.2.5.3 Kit 2 Process Improvement Proposal.

1.3.2.9.8 Kit 6 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.9.9 Kit 6 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.2.10 Kit 7.

1.3.2.10.1 Kit 7 Main Page. The Kit 7 Main page has two frames on it. The frame in the left provides the same navigation link with the Welcome Page. The frame in the right shall describe the requirements for each assignment.

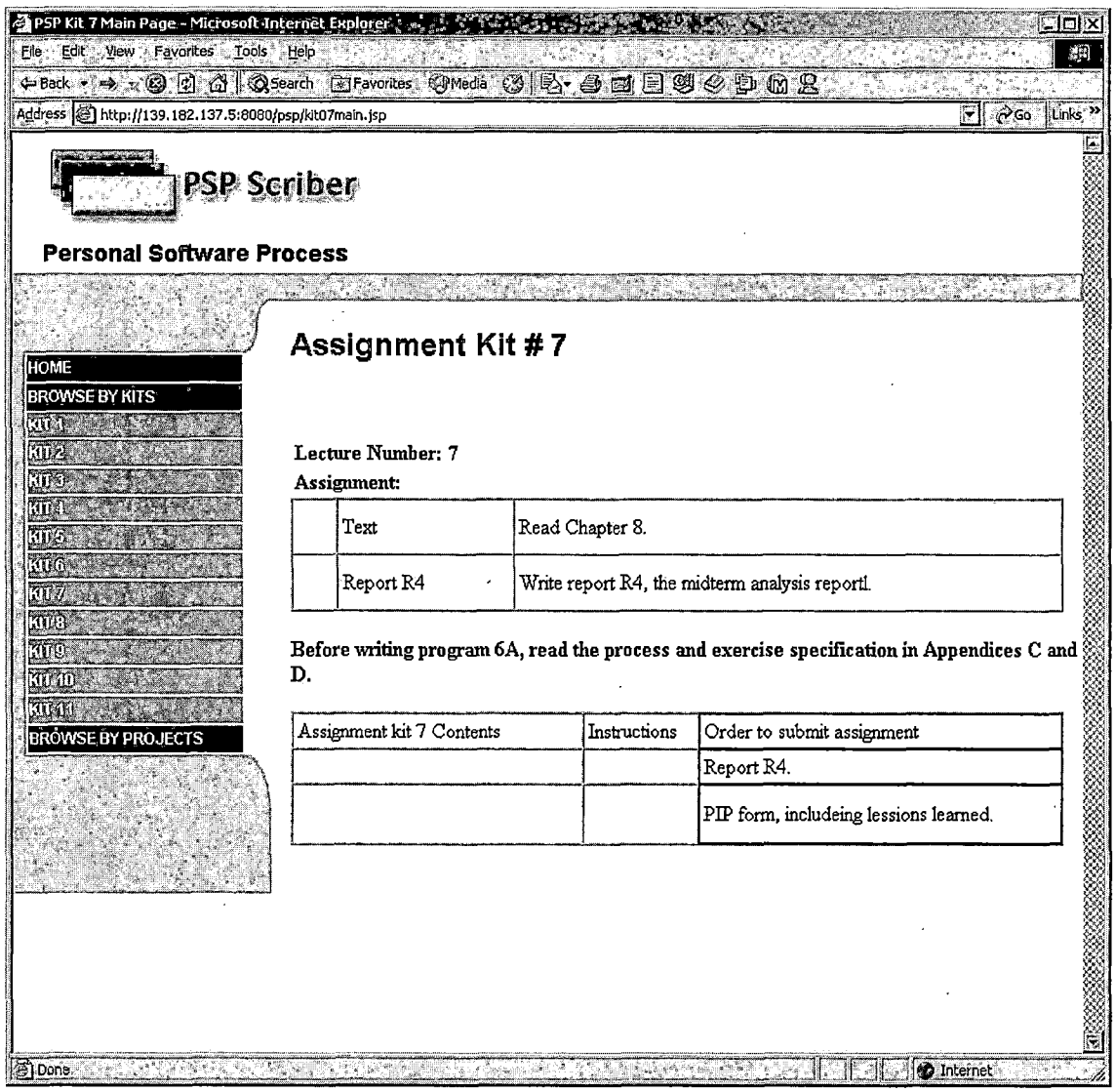


Figure 35. Screen Shot of Kit 7 Main Page

1.3.2.11 Kit 8.

1.3.2.11.1 Kit 8 Main Page. The Kit 8 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and

standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

The differences between the kit 8 and the kit 6 are the kit 8 uses the PSP2 Project Plan Summary instead of using the PSP1.1 Project Plan Summary.

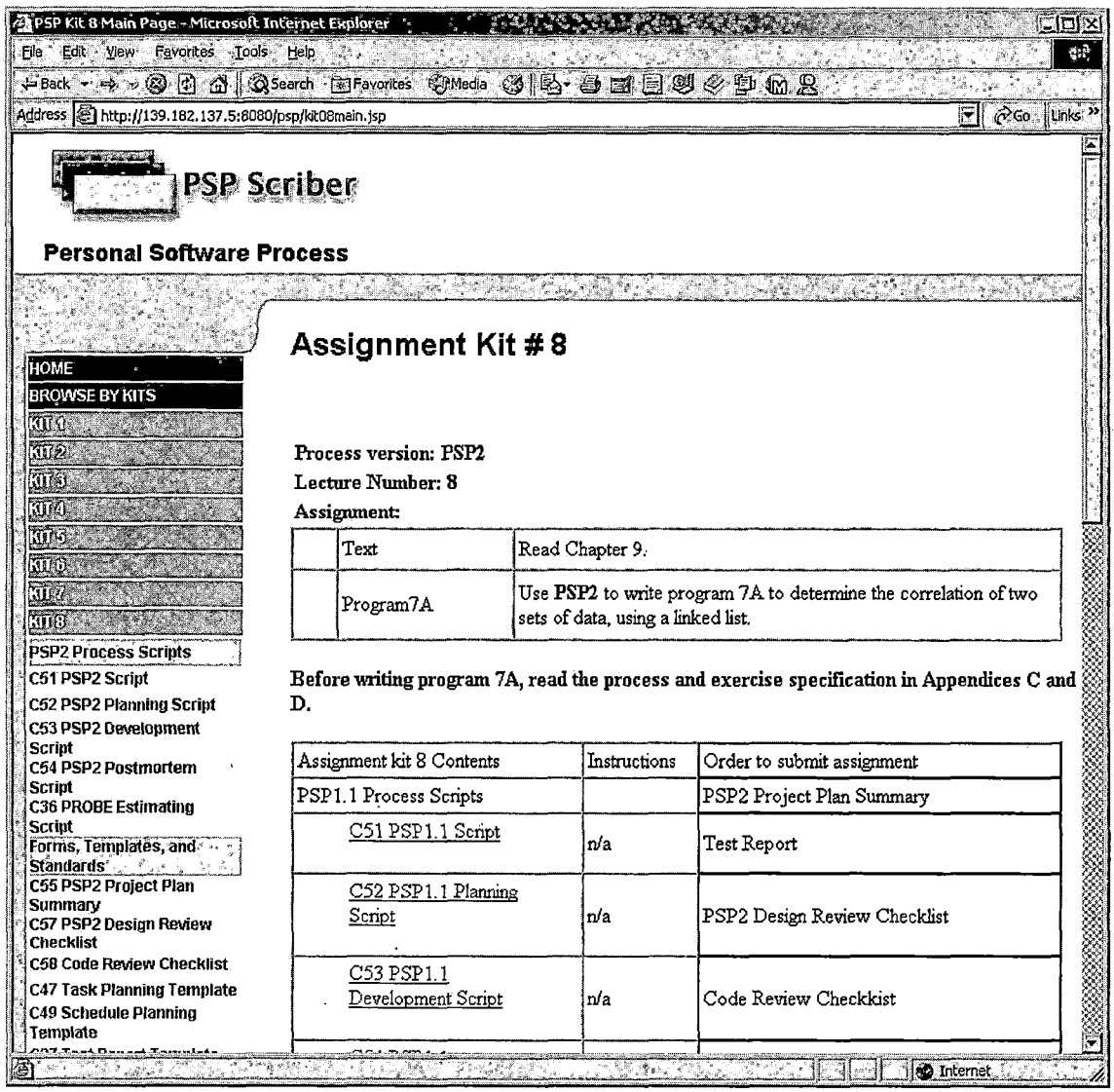


Figure 36. Screen Shot of Kit 8 Main Page

1.3.2.11.2 Kit 8 PSP2 Project Plan Summary. The PSP2 Project Plan Summary also has two frames on it. The left frame provides the same function as Main Page. The Left frame provides the PSP applet to show PSP2 Project Plan Summary. User can enter data on this PSP applet. The

applet will perform the calculate function (e.g. to date planning time) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

PSP2.0: Project Plan Summary - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address http://139.182.137.5:8080/psp/projplan20applet.jsp?kit=navibars/navikit8.html

PSP Scriber

Personal Software Process

HOME
BROWSE BY KITS
KIT 1
KIT 2
KIT 3
KIT 4
KIT 5
KIT 6
KIT 7
KIT 8
PSP2 Process Scripts
C51 PSP2 Script
C52 PSP2 Planning Script
C53 PSP2 Development Script
C54 PSP2 Postmortem Script
C36 PROBE Estimating Script
Forms, Templates, and Standards
C55 PSP2 Project Plan Summary
C57 PSP2 Design Review Checklist
C58 Code Review Checklist
C47 Task Planning Template
C49 Schedule Planning Template

PSP2 Project Plan Summary

Employee	Heng-Jui Tsao	Date	1999-11-22
Project Name	PSP Scriber Test Program 8A	PSP Progress	2.0
Supervisor	Dr. A. I. Concepcion	Language	C++

Summary	Plan	Actual	To Date
LOC/Hour	21.677	36.936	21.339
Planned Time	310		1288
Actual Time		346	2002
CPI(Cost Performance Index)			0.643 (Planned/Actual)
% Reused	0.000	0.000	6.039
% New Reused	0.000	0.000	14.524
Test Defects/KLOC	55.046	29.851	45.238
Total Defects/KLOC	137.615	104.478	128.571
Yield %	0.000	0.000	0.000

Program Size (LOC):	Plan	Actual	To Date
Base (B)	88 (Measured)	88 (Measured)	
Deleted (D)	5 (Estimated)	2 (Counted)	
Modified (M)	10 (Estimated)	7 (Counted)	
Added (A)	29 (N-M)	127 (T-B+D-R)	
Reused (R)	0 (Estimated)	0 (Counted)	43
Total New & Changed (N)	109	134	420

Applet started. Internet

Figure 37. Screen Shot of PSP2 Project Planning Summary

1.3.2.11.3 Kit 8 Task Planning Template. As shown on 1.3.2.8.3 Kit 5 Task Planning Template.

1.3.2.11.4 Kit 8 Schedule Planning Template. As shown on 1.3.2.8.4 Kit 5 schedule Planning Template.

1.3.2.11.5 Kit 8 Test Report Template. As shown on 1.3.2.7.3 Kit 4 Test Report Template.

1.3.2.11.6 Kit 8 Size Estimating Template. As shown on 1.3.2.7.4 Kit 4 Size Estimating Template.

1.3.2.11.7 Kit 8 Process Improvement Proposal. As shown on 1.3.2.5.3 kit 2 Process Improvement Proposal.

1.3.2.11.8 Kit 8 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.11.9 Kit 8 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.2.12 Kit 9.

1.3.2.12.1 Kit 9 Main Page. The Kit 9 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame on the right shall describe the requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame

provides the instruction link for the forms this kit will need.

The kit 9 has all the forms and templates as same as the kit 8.

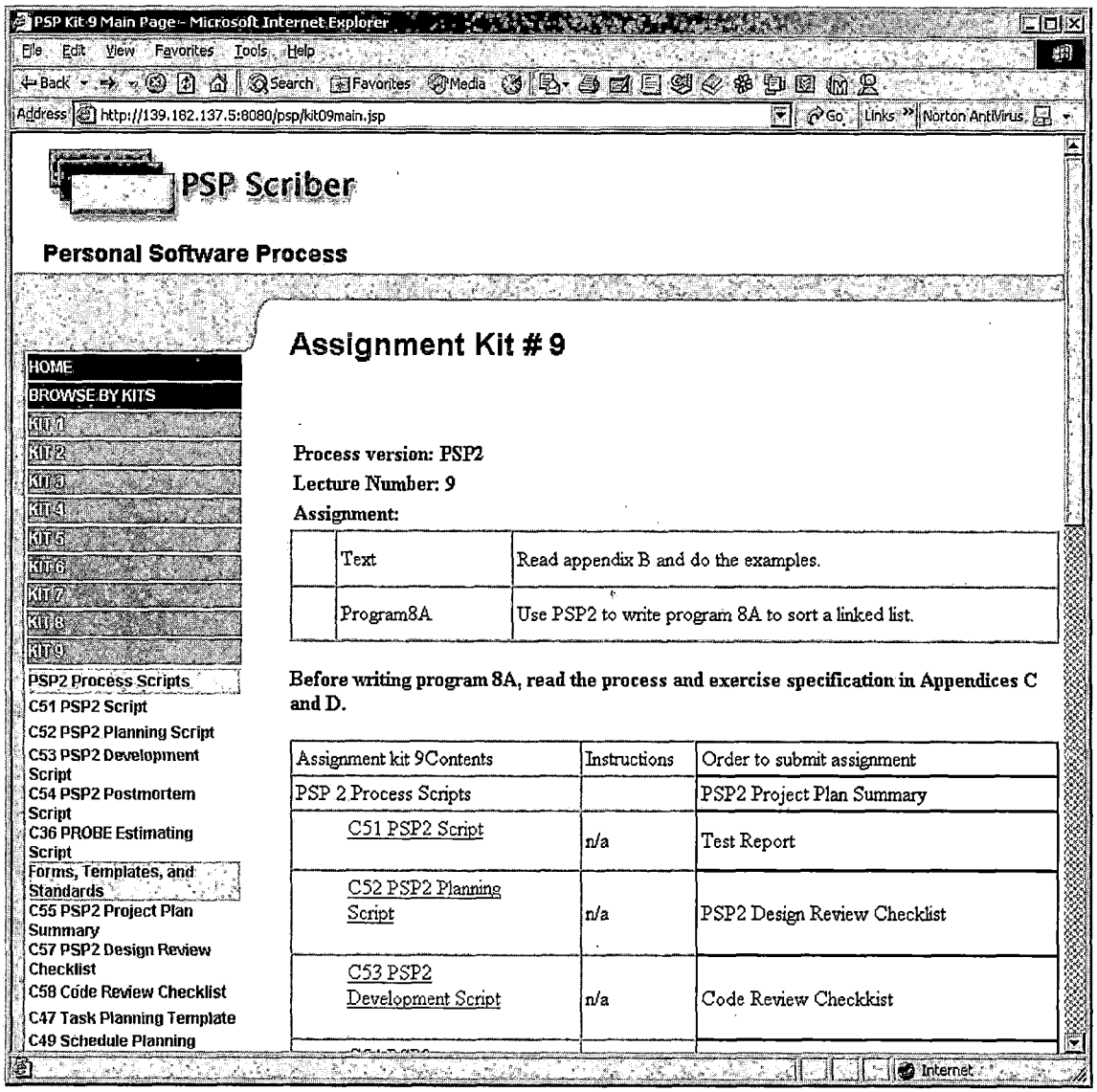


Figure 38. Screen Shot of Kit 9 Main Page

1.3.2.12.2 Kit 9 PSP2 Project Plan Summary. As shown on 1.3.2.11.2 Kit 8 PSP2 Project Plan Summary.

1.3.2.12.3 Kit 9 Task Planning Template. As shown on 1.3.2.8.3 Kit 5 Task Planning Template.

1.3.2.12.4 Kit 9 Schedule Planning Template. As shown on 1.3.2.8.4 Kit 5 schedule Planning Template.

1.3.2.12.5 Kit 9 Test Report Template. As shown on 1.3.2.7.3 Kit 4 Test Report Template.

1.3.2.12.6 Kit 9 Size Estimating Template. As shown on 1.3.2.7.4 Kit 4 Size Estimating Template.

1.3.2.12.7 Kit 9 Process Improvement Proposal. As shown on 1.3.2.5.3 Kit 2 Process Improvement Proposal.

1.3.2.12.8 Kit 9 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.12.9 Kit 9 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.2.13 Kit 10.

1.3.2.13.1 Kit 10 Main Page. The Kit 10 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame in the right shall describe the

requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit will need.

The differences between the kit 10 and the kit 9 are the kit 10 uses the PSP2.1 Project Plan Summary instead of using the PSP2 Project Plan Summary. Besides, instead of using Object-Oriented approach and design process created by Humphrey, PSP Scriber use UML use case diagram, class diagram, sequential diagram to perform Object-Oriented approach.

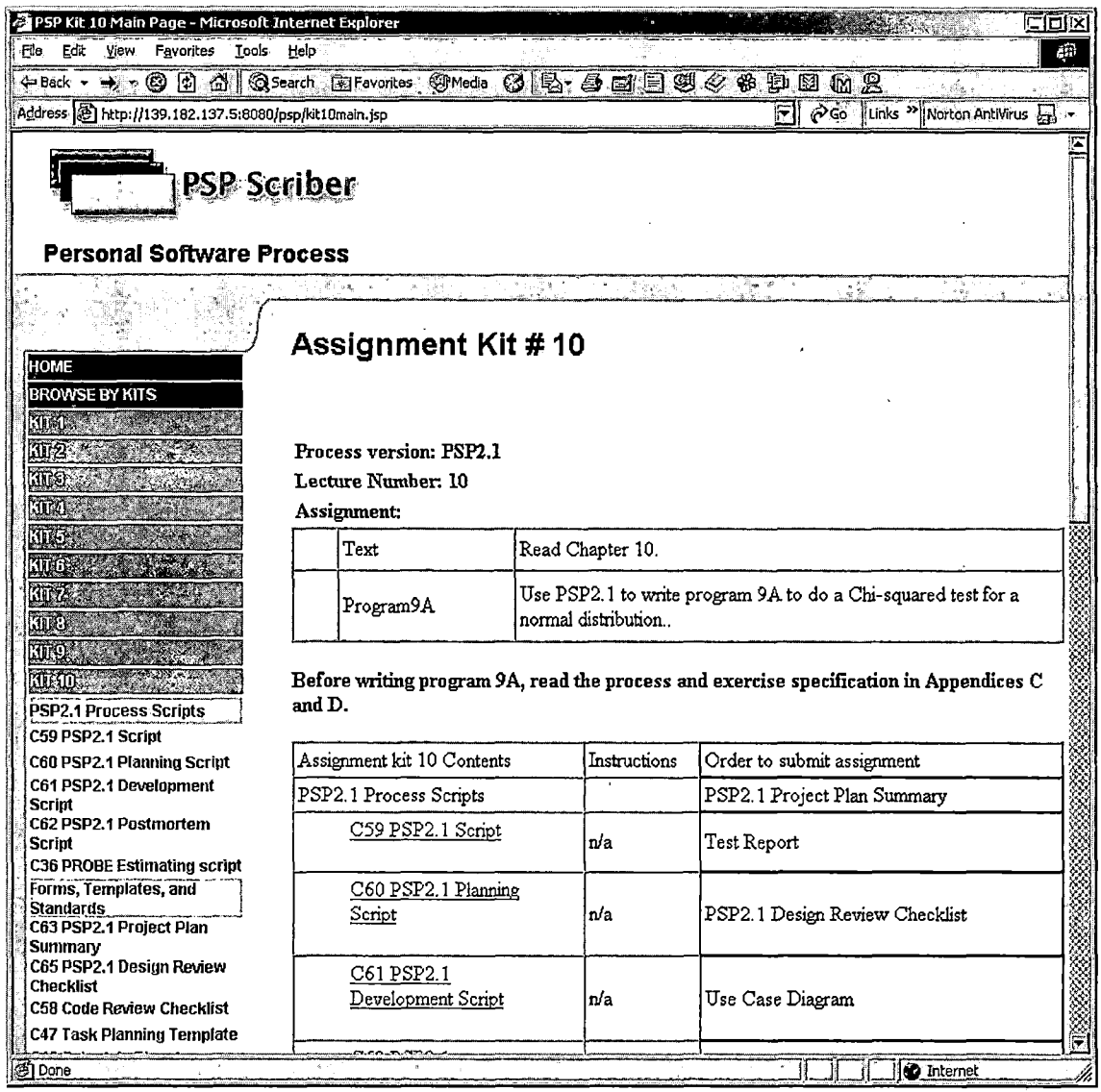


Figure 39. Screen Shot of Kit 10 Main Page

1.3.2.13.2 Kit 10 PSP2.1 Project Plan Summary.

The PSP2.1 Project Plan Summary also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the PSP2.1 Project Plan Summary. The user can enter data on this PSP

applet. The applet will perform the calculate function (e.g. to date planning time) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

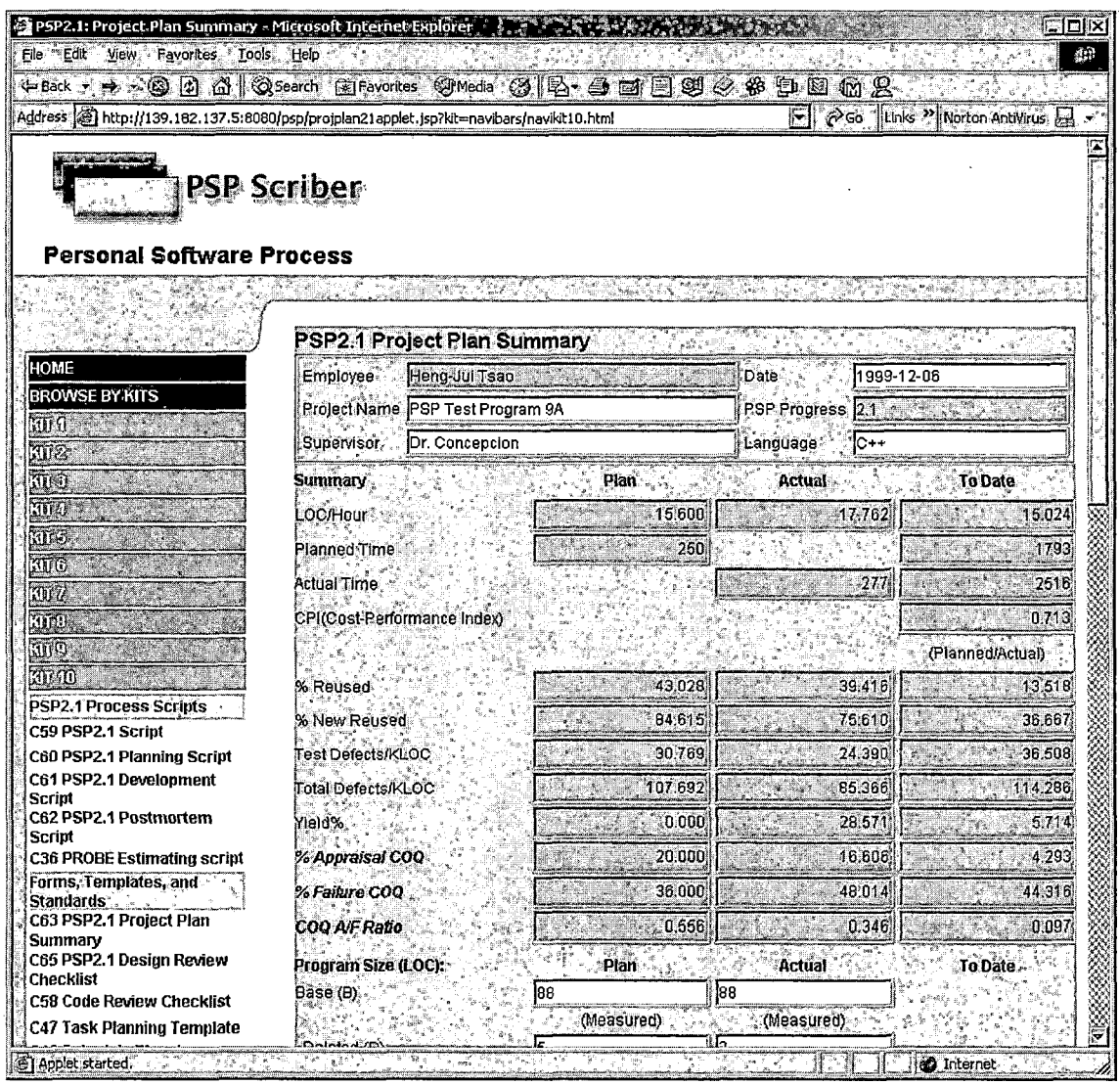


Figure 40. Screen Shot of Kit 10 PSP2.1 Project Planning Summary

1.3.2.13.3 Kit 10 Task Planning Template. As shown on 1.3.2.8.3 Kit 5 Task Planning Template.

1.3.2.13.4 Kit 10 Schedule Planning Template. As shown on 1.3.2.8.4 Kit 5 schedule Planning Template.

1.3.2.13.5 Kit 10 Test Report Template. As shown on 1.3.2.7.3 Kit 4 Test Report Template.

1.3.2.13.6 Kit 10 Size Estimating Template. As shown on 1.3.2.7.4 Kit 4 Size Estimating Template.

1.3.2.13.7 Kit 10 Process Improvement Proposal. As shown on 1.3.2.5.3 Kit 2 Process Improvement Proposal.

1.3.2.13.8 Kit 10 Time Recording Log. As shown on 1.3.2.4.3 kit 1 Time Recording Log.

1.3.2.13.9 Kit 10 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.2.14 Kit 11.

1.3.2.14.1 Iteration Page. The PSP3 introduces an iteration process design to help you to development larger programs. User can click a particular iteration he/she wants to work. Or, user can click insert button for inserting a new iteration.

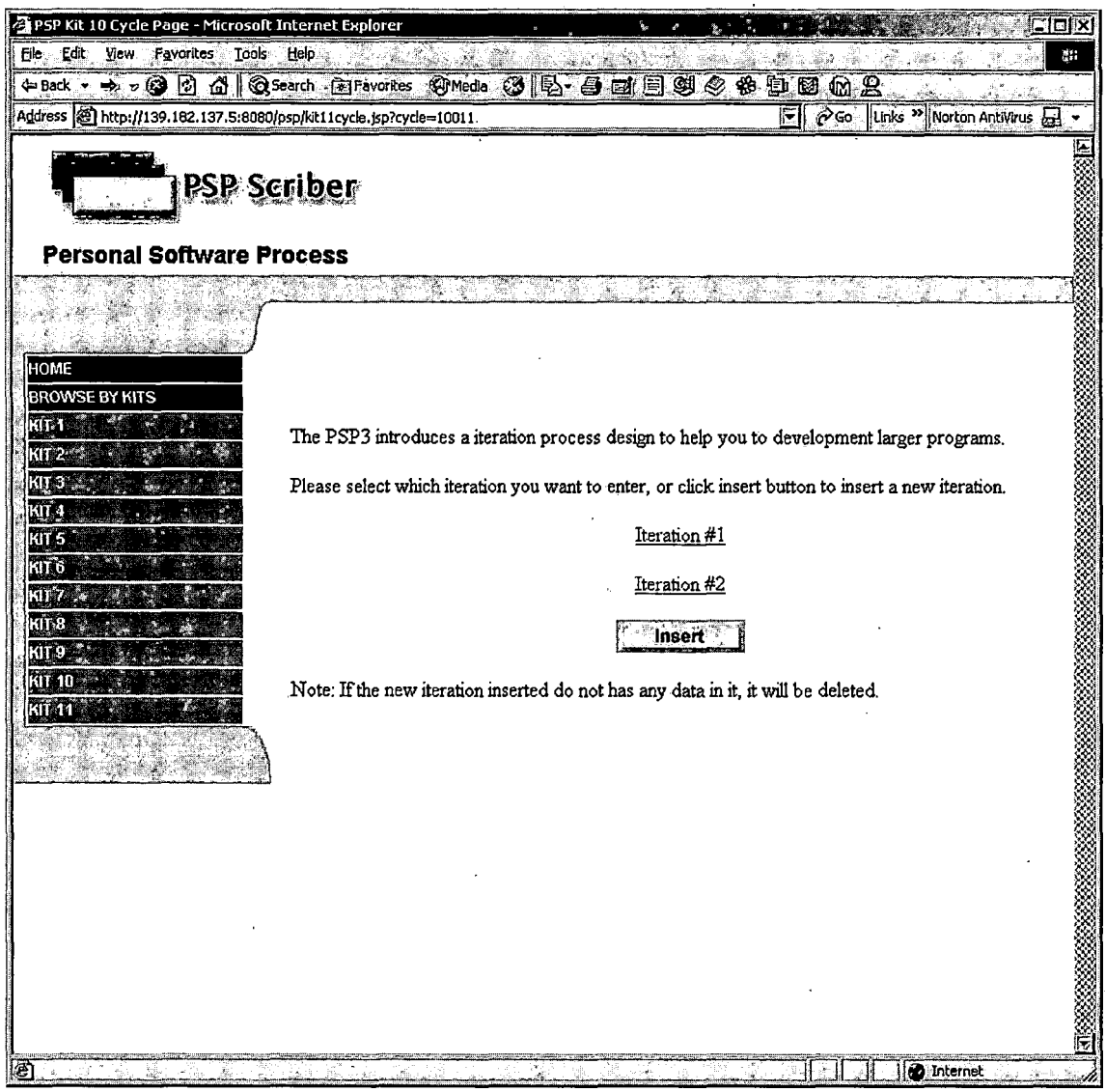


Figure 41. PSP3 Iteration Page

1.3.2.14.2 Kit 11 Main Page. The Kit 11 Main page has two frames on it. The frame on the left provides the same navigation link with Welcome Page. In addition, it also provides links for scripts, forms, templates, and standards. The frame on the right shall describe the

requirements for each assignment, and provides links for scripts, forms, templates, and standards. Also this frame provides the instruction link for the forms this kit is needed.

The differences between the kit 11 and the kit 10 are the kit 11 uses the PSP3 Project Plan Summary instead of using the PSP2.1 Project Plan Summary.

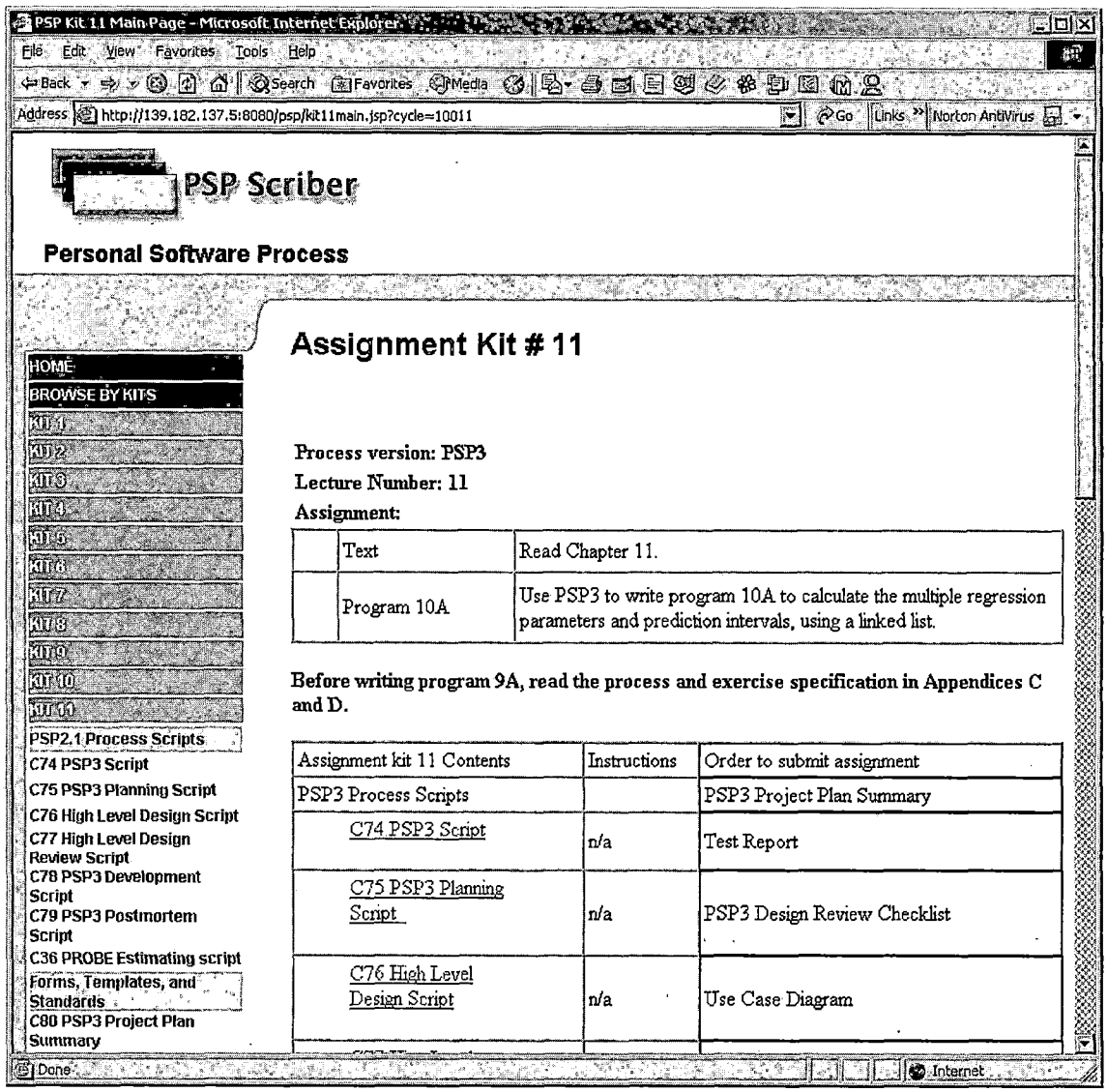


Figure 42. Screen Shot of Kit 11 Main Page

1.3.2.14.3 Kit 11 PSP3 Project Plan Summary. The PSP3 Project Plan Summary also has two frames on it. The left frame provides the same function as the Main Page. The Left frame provides the PSP applet to show the PSP3

Project Plan Summary. The user can enter data on this PSP applet. The applet will perform the calculate function (e.g. to date planning time) defined by Humphrey, and save data into PSP database. The left frame also provides a print button to allow user print out this form with data on it.

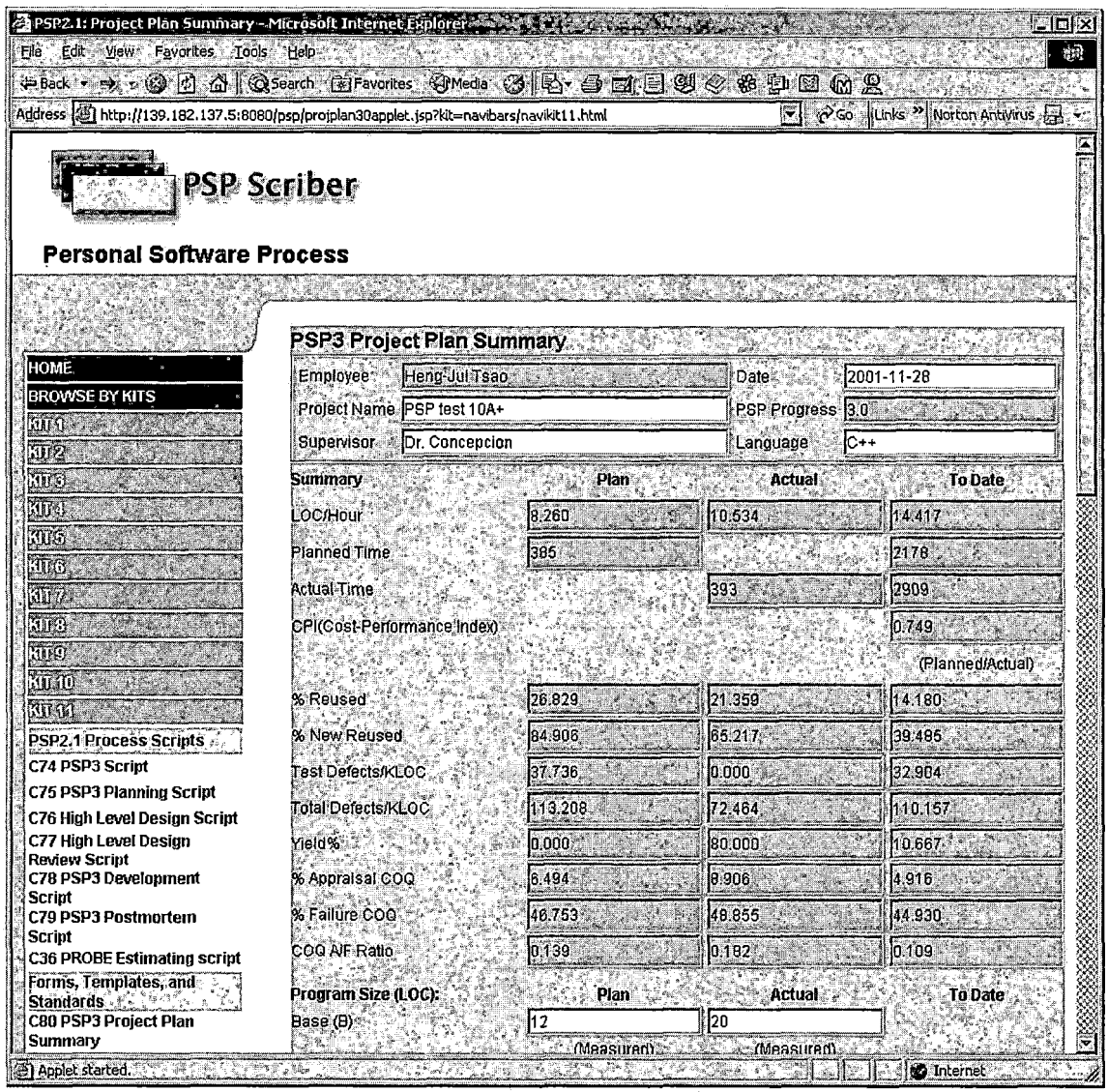


Figure 43. Screen Shot of PSP3 Project Planning Summary

1.3.2.14.4 Kit 11 Task Planning Template. As shown on 1.3.2.8.3 Kit 5 Task Planning Template.

1.3.2.14.5 Kit 11 Schedule Planning Template. As shown on 1.3.2.8.4 Kit 5 Schedule Planning Template.

1.3.2.14.6 Kit 11 Test Report Template. As shown on 1.3.2.7.3 Kit 4 Test Report Template.

1.3.2.14.7 Kit 11 Size Estimating Template. As shown on 1.3.2.7.4 Kit 4 Size Estimating Template.

1.3.2.14.8 Kit 11 Process Improvement Proposal. As shown on 1.3.2.5.3 Kit 2 Process Improvement Proposal.

1.3.2.14.9 Kit 11 Time Recording Log. As shown on 1.3.2.4.3 Kit 1 Time Recording Log.

1.3.2.14.10 Kit 11 Defect Recording Log. As shown on 1.3.2.4.4 Kit 1 Defect Recording Log.

1.3.3 Performance Requirements

This is not a time critical application, so performance will be defined as that which the system is able to provide. To access all PSP forms, scripts and their respective instructions, the machine will be required to have a Web browser. To access PSP Applet, the machine will be required to support Sun Java 2 and the Swing Library, as well as Internet Explorer 4.01 or greater, or Netscape Navigator 4.0 or greater, or a comparable browser.

1.3.4 Logical Database Requirements

PSP Scriber will store all the data in a DBMS, such as Oracle 8. The class diagram and ER diagram of the database system for PSP Scriber is showing following.

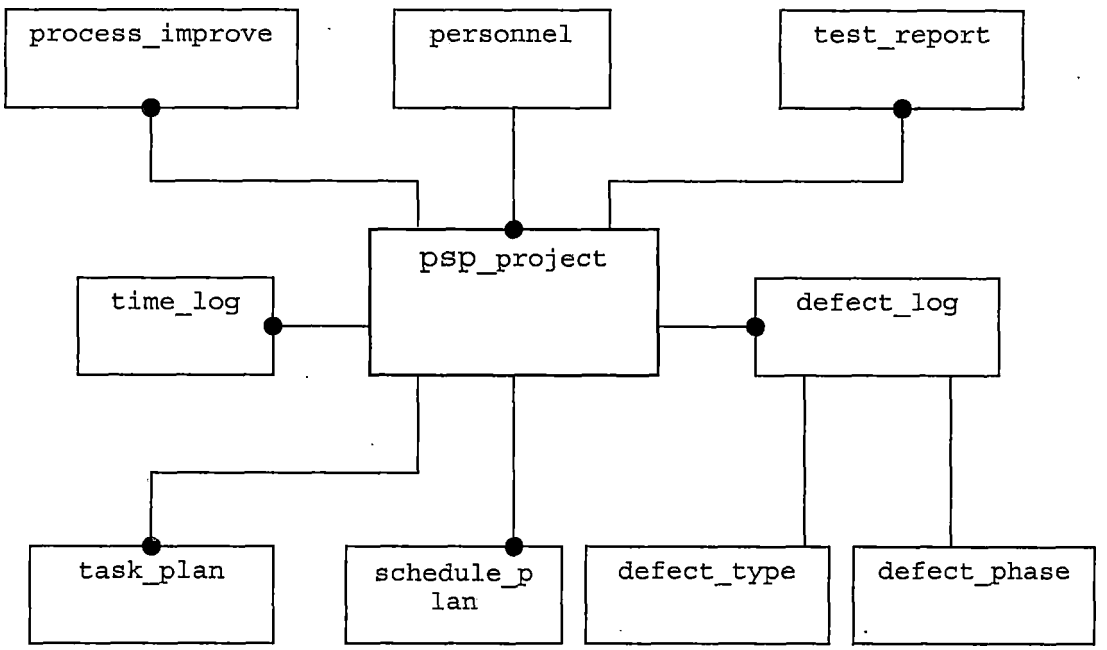


Figure 44. PSP Scriber Database Class Diagram

1.3.5 Design Constraints

The contents covered in the user interface will not exceed PSP forms, scripts and their respective instructions in the textbook, which is listed in the Reference section.

1.3.6 Software System Attributes

1.3.6.1 Reliability. Described in section 1.2.

1.3.6.2 Availability. The accessible PSP forms, documentation, instructions and examples will be available at a published web address. The software will remain available at the published web address except when system administration technical problems deny service to this published web address.

1.3.6.3 Security. PSP Scriber will be freely available for use. Web pages themselves are secured by the host machines security systems, and are reasonably safe from tampering by the general public. PSP Scriber will support different user access privileges and protect the software engineer's historical data. Only the administrator has security privileges to change the software engineer's historical data.

1.3.6.4 Maintainability. The majority of PSP Scriber is implemented as HTML pages, and are easily maintained using a web development environment such as FrontPage 2000.

PSP Applet is a Java-based application that relies on object-oriented design and implementation to facilitate future maintenance and expansion.

1.3.6.5 Portability. PSP Scriber will be designed to meet or exceed the abilities of many current Internet browser technologies. It will be cross platform compatible and expected that browser technologies will progress at the same rate as they have in the past, which would enable further platform availability.

CHAPTER TWO

SOFTWARE DESIGN

2.1 Architecture

The goal of this project is to produce a Web-based software engineering tool which contain four major parts: HTML forms, the Java Bean based JDBC classes, the Java based applets, and PSP database.

This section concentrates on the architecture representation that leads to a detailed PSP database class diagram and PSP applet class diagram

2.1.1 JDBC Classes Class Diagram

The JDBC classes perform the connection between the PSP applets and PSP database. It makes it easy to send SQL statement to the relational database system and send back the result to PSP applets. Also, the JDBC classes accept the updated data, which is from PSP applet, then send it back to PSP database. Furthermore, the Java Bean based JDBC classes perform the basic mathematical calculation inside the PSP applet and calculate the result for the JSP pages.

For JDBC classes, the JDBCDriver is the parent class for other JDBC classes, which are of the classes named ending in "JDBC".

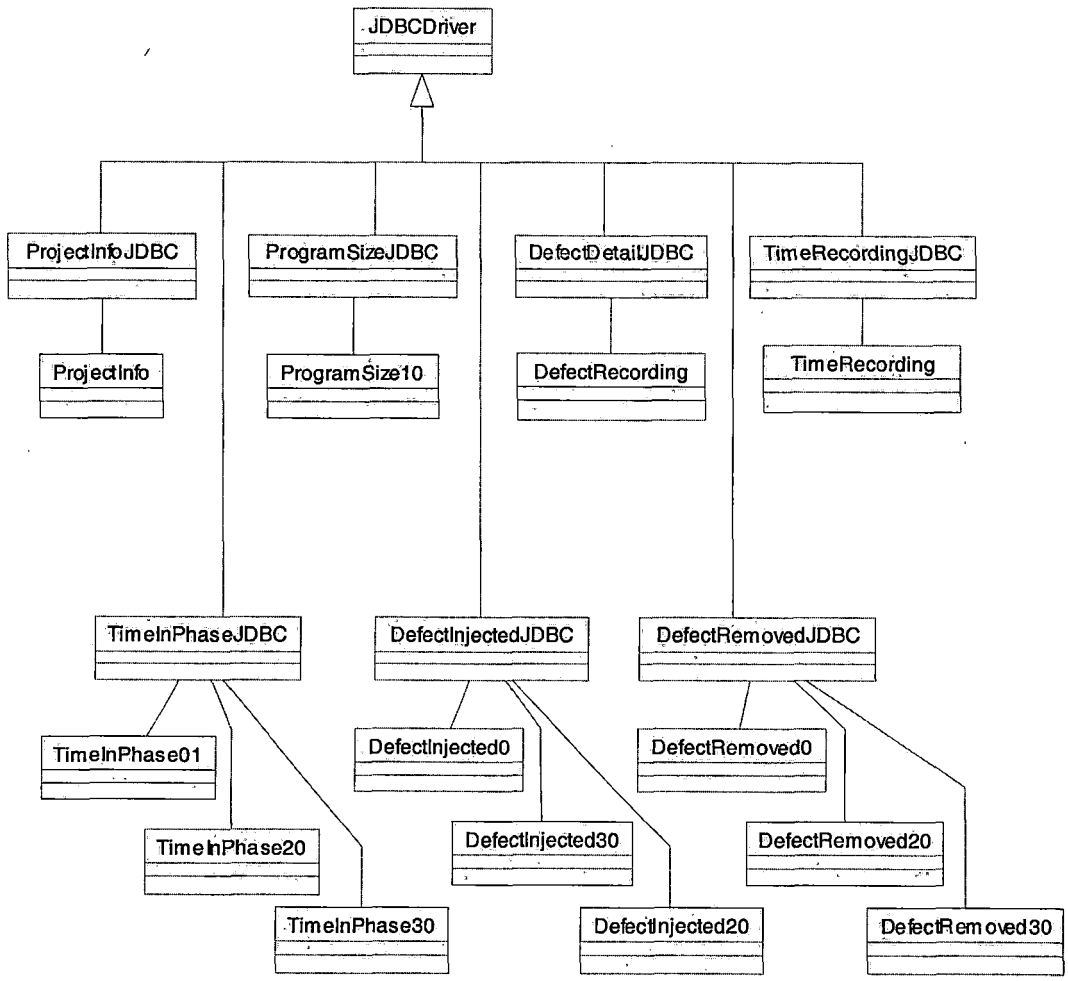


Figure 45. JDBC Classes Class Diagram

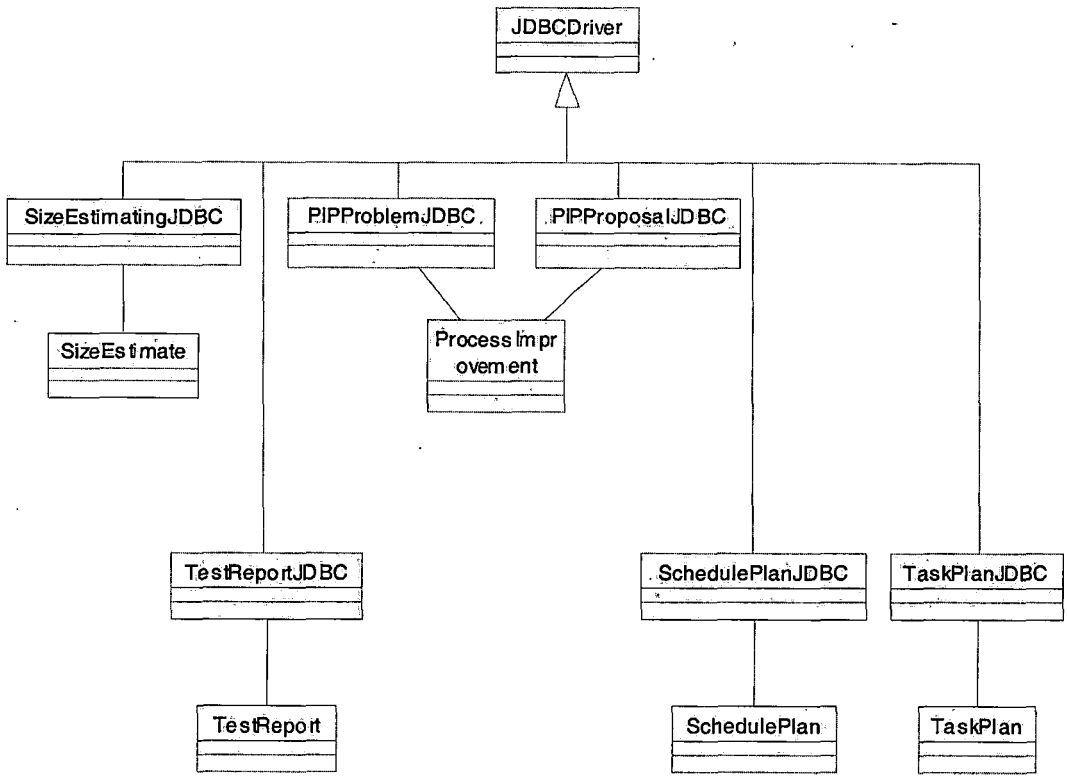


Figure 45: (Continued)

2.1.2 Applets Class Diagram

The PSP applet components of PSP Scriber rely on Java applet technology to provide navigation, display, calculations, text and image rendering. The PSP applet will perform the basic database access from the JDBC class. Additionally, the PSP applet will provide the data input and update for the PSP database. Once the data is entered into the PSP applet, the PSP applet will call Java Bean based JDBC class and perform the database connection.

The PSP applets separate two parts of the form in their usage. The first one is Java Bean-based components, which use the JDBC classes as their database connection and perform the calculate function as defined by Humphrey. The Java Bean-based components are ProjectInfo, ProgramSize01, TimeInPhase01, TimeInPhase20, TimeInPhase30, DefectInject0, DefectInject20, DefectInject30, defectRemoved0, DefectRemoved20, and DefectRemoved30.

The second part is Java applets, which grab the Java Bean base components and perform the other necessary functions to display on the Web browser. The Java Applets are ProjectPlan00, ProjectPlan01, ProjectPlan10, ProjectPlan11, ProjectPlan20, ProjectPlan21, ProejctPlan30, TimeRecording, DefectRecording, ProcessImprove, TestReport, SizeEstimating, TaskPlan, and SchedulePlan.

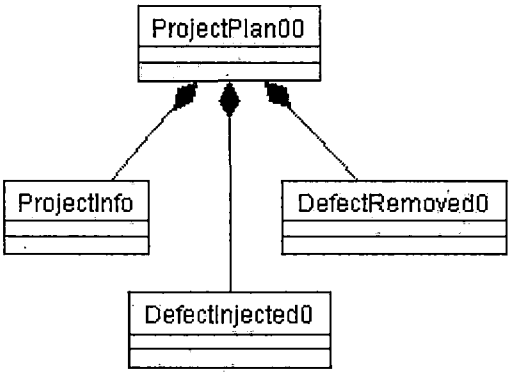


Figure 46. ProjectPlan00 Class Diagram

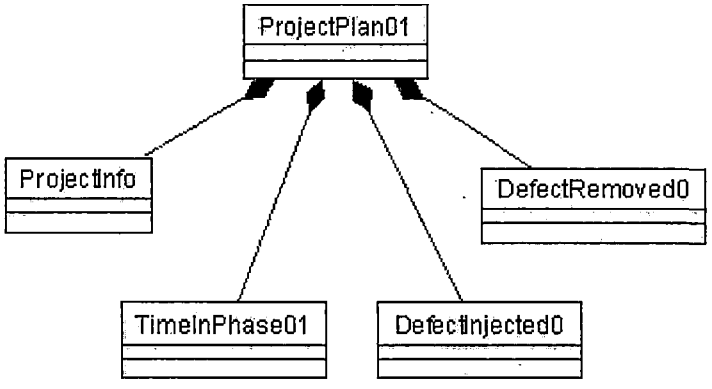


Figure 47. ProjectPlan01 Class Diagram

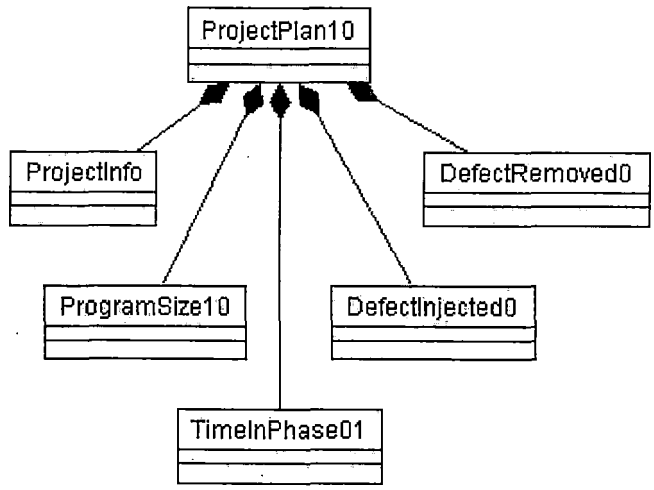


Figure 48. ProjectPlan10 Class Diagram

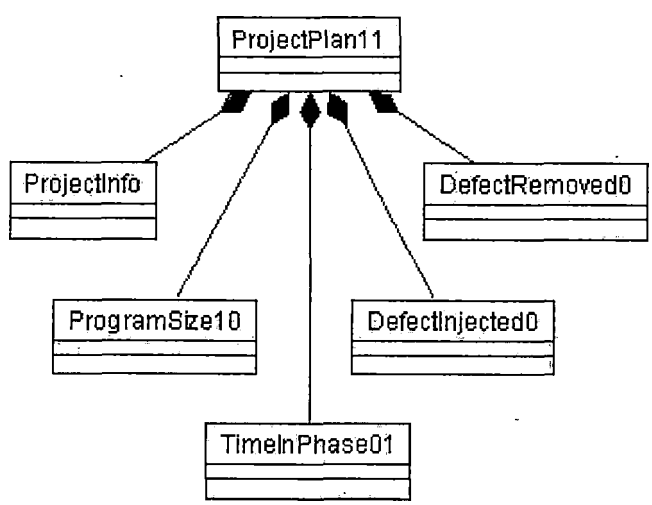


Figure 49. ProjectPlan11 Class Diagram

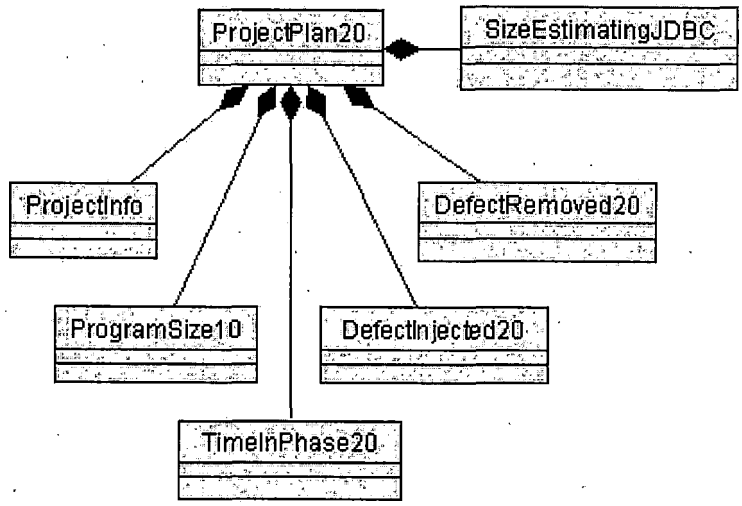


Figure 50. ProjectPlan20 Class Diagram

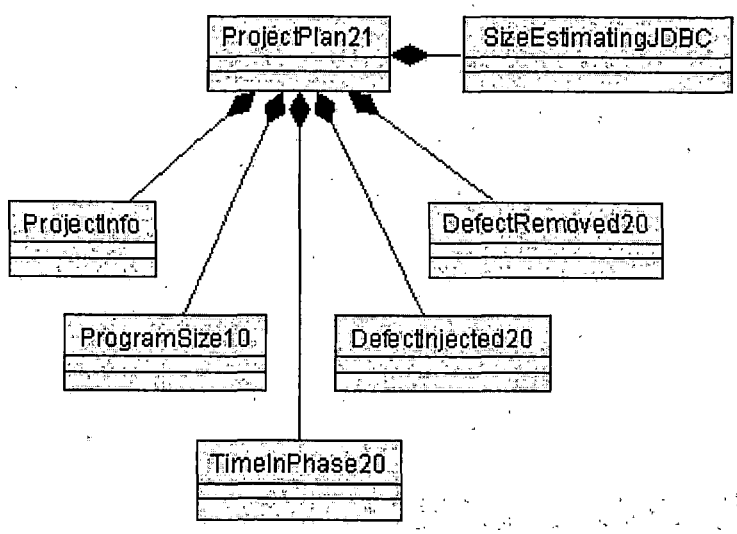


Figure 51. ProjectPlan21 Class Diagram

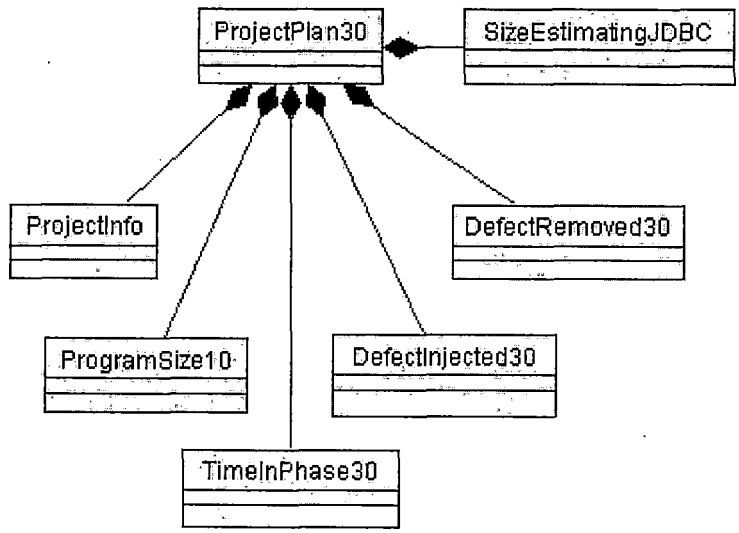


Figure 52. ProjectPlan30 Class Diagram

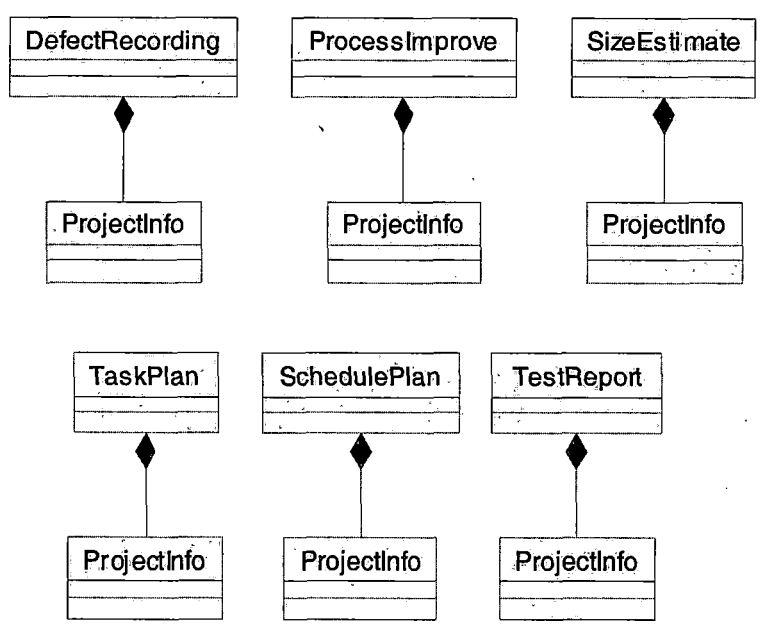


Figure 53. Other Applets Class Diagram

2.2 Detailed Design

This section concentrates on the refinements to the architecture representation that lead to detailed data structure and algorithmic representation for the PSP Scriber. The following sub-sections are organized by Java classes.

2.2.1 JDBCDriver

The purpose of JDBCDriver class is to provide the database management system driver and perform the connection and disconnection from PSP database for the usage of other JDBC class.

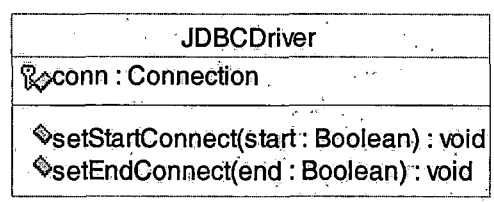


Figure 54. JDBCDriver Detailed Class Diagram

2.2.2 ProjectInfoJDBC

The purpose of ProjectInfo JDBC is to access and update project name, employee name, supervisor, starting date of the project, PSP progress, and programming language from PSP database.

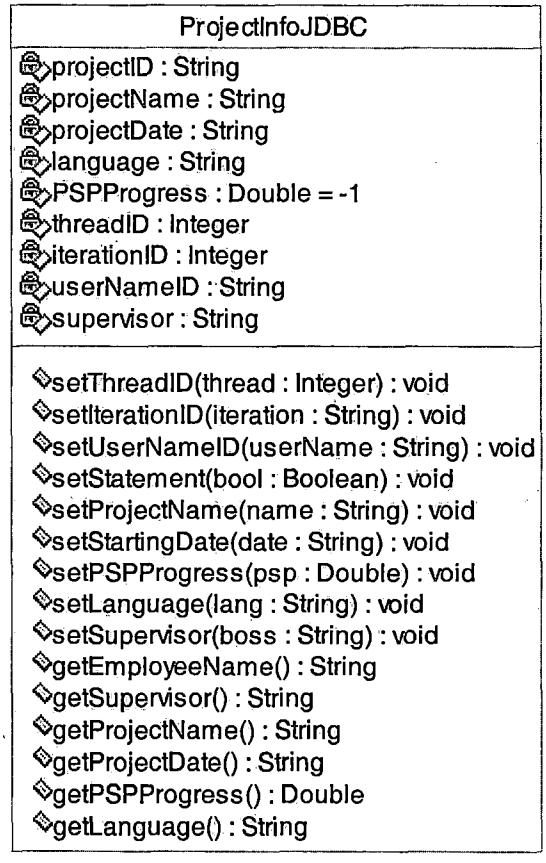


Figure 55. ProjectInfoJDBC Detail Class Diagram

2.2.3 ProgramSizeJDBC

The purpose of ProgramSizeJDBC class is to access and update PSP database for use by the ProgramSize10 class. It accesses and updates plan, actual, and to date lines of code. It includes the following data items: base, deleted, modified, added, reused, new and changed, total, and new reused.

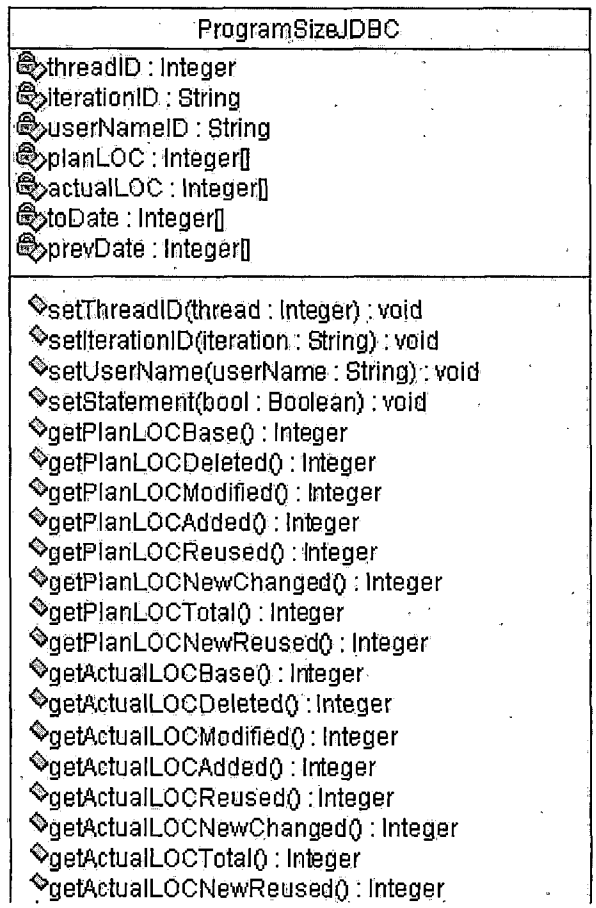


Figure 56. ProgramSizeJDBC Detailed Class Diagram

getToDateReused() : Integer
getToDateNewChanged() : Integer
getToDateTotal() : Integer
getToDateNewReused() : Integer
getPrevToDateReused() : Integer
getPrevToDateNewChanged() : Integer
getPrevToDateTotal() : Integer
getPrevToDateNewReused() : Integer
setPlanLOCBase(LOC : Integer) : void
setPlanLOCDeleted(LOC : Integer) : void
setPlanLOCModified(LOC : Integer) : void
setPlanLOCAdded(bool : Boolean) : void
setPlanLOCReused(LOC : Integer) : void
setPlanLOCNewChanged(LOC : Integer) : void
setPlanLOCTotal(bool : Boolean) : void
setPlanLOCNewReused(LOC : Integer) : void
setActualLOCBase(LOC : Integer) : void
setActualLOCDeleted(LOC : Integer) : void
setActualLOCModified(LOC : Integer) : void
setActualLOCAdded(bool : Boolean) : void
setActulaLOCReused(LOC : Integer) : void
setActualLOCNewChanged(bool : Boolean) : void
setActualLOCTotal(LOC : Integer) : void
setActualLOCNewReused(LOC : Integer) : void
setAllLOC(bool : Boolean) : void

Figure 56. (Continued)

2.2.4 DefectDetailJDBC Class

The purpose of DefectDetailJDBC class is to access and update PSP database for use by the DefectRecording class.

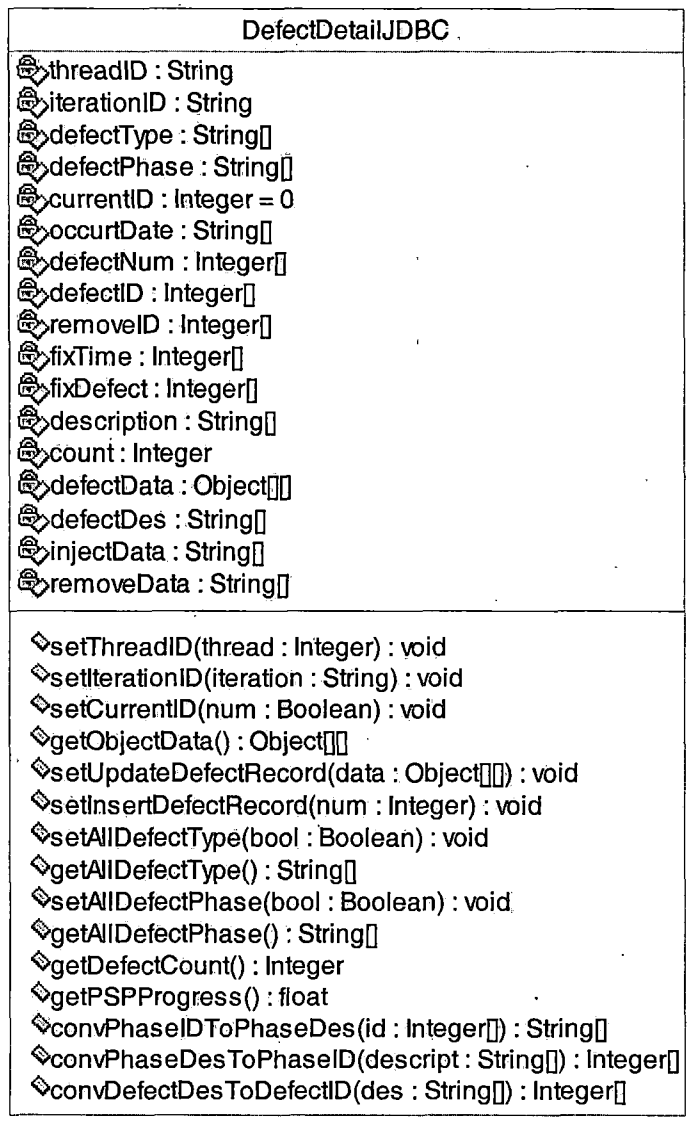


Figure 57. DefectDetailJDBC Detailed Class Diagram

2.2.5 TimeInPhaseJDBC Class

The purpose of TimeInPhaseJDBC class is to access and update PSP database for use by the TimeInPhase class. It accesses and updates plan time, actual time, to date time,

and to date percentage. It includes the following data items: planning, high level design, high level design review, detail design, detail design review, code, code review, compile, test, postmortem, and total.

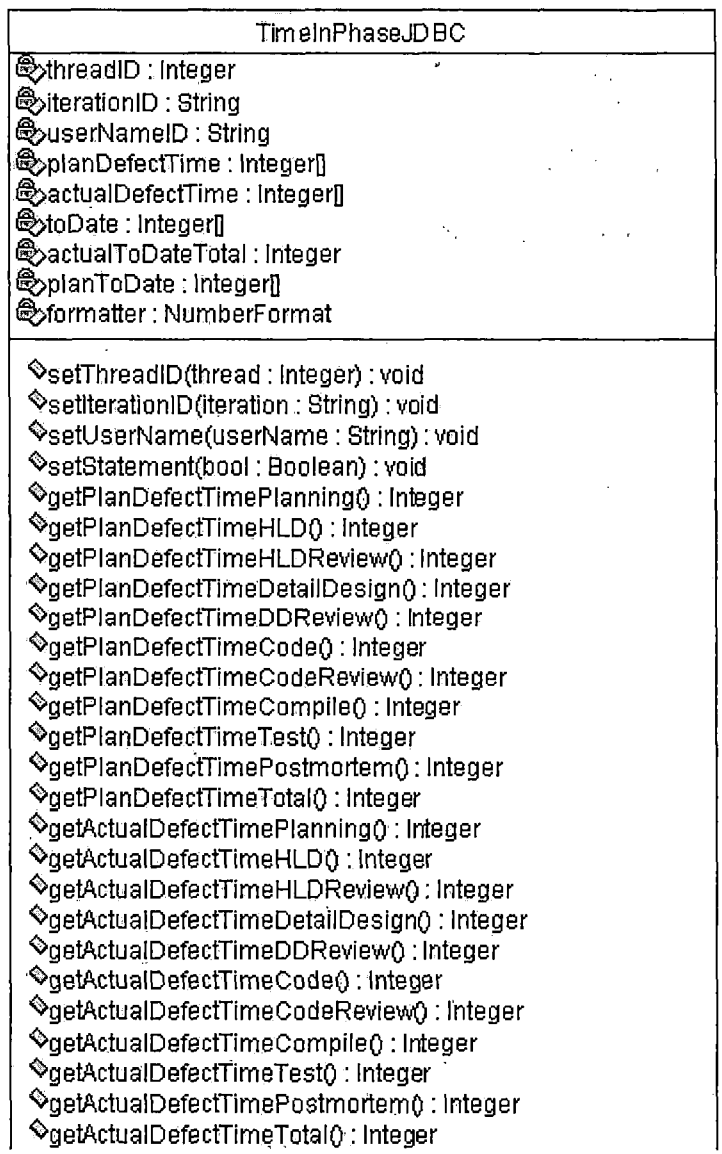


Figure 58. TimeInPhaseJDBC Detailed Class Diagram

getToDateDefectTimePlanning() : Integer
getToDateDefectTimeHLD() : Integer
getToDateDefectTimeHLDReview() : Integer
getToDateDefectTimeDetailDesign() : Integer
getToDateDefectTimeDDReview() : Integer
getToDateDefectTimeCode() : Integer
getToDateDefectTimeCodeReview() : Integer
getToDateDefectTimeCompile() : Integer
getToDateDefectTimeTest() : Integer
getToDateDefectTimePostmortem() : Integer
getToDateDefectTimeTotal() : Integer
getToDatePerDefectTimePlanning() : String
getToDatePerDefectTimeHLD() : String
getToDatePerDefectTimeHLDReview() : String
getToDatePerDefectTimeDetailDesign() : String
getToDatePerDefectTimeDDReview() : String
getToDatePerDefectTimeCode() : String
getToDatePerDefectTimeCodeReview() : String
getToDatePerDefectTimeCompile() : String
getToDatePerDefectTimeTest() : String
getToDatePerDefectTimePostmortem() : String
getToDatePerDefectTimeTotal() : String
getPlanToDateDefectTimeTotal() : Integer
setPlanDefectTimePlanning(defectTime : Integer) : void
setPlanDefectTimeHLD(defectTime : Integer) : void
setPlanDefectTimeHLDReview(defectTime : Integer) : void
setPlanDefectTimeDetailDesign(defectTime : Integer) : void
setPlanDefectTimeDDreview(defectTime : Integer) : void
setPlanDefectTimeCode(defectTime : Integer) : void
setPlanDefectTimeCodeReview(defectTime : Integer) : void
setPlanDefectTimeCompile(defectTime : Integer) : void
setPlanDefectTimeTest(defectTime : Integer) : void
setPlanDefectTimePostmortem(defectTime : Integer) : void
setActualDefectTimePlanning(defectTime : Integer) : void
setActualDefectTimeHLD(defectTime : Integer) : void
setActualDefectTimeHLDReview(defectTime : Integer) : void
setActualDefectTimeDetailDesign(defectTime : Integer) : void
setActualDefectTimeDDReview(defectTime : Integer) : void
setActualDefectTimeCode(defectTime : Integer) : void
setActualDefectTimeCodeReview(defectTime : Integer) : void
setActualDefectTimeCompile(defectTime : Integer) : void
setActualDefectTimeTest(defectTime : Integer) : void
setActualDefectTimePostmortem(defectTime : Integer) : void
setAllDefectTime(bool : Boolean) : void

Figure 58. (Continued)

2.2.6 DefectInjectedJDBC Class

The purpose of DefectInjectedJDBC class is to access and update the PSP database for use by the DefectInjected class. It accesses and updates the number of defects injected in plan, actual, to date, and the to date percentage. It includes these major items: planning, high level design, high level design review, detail design, detail design review, code, code review, compile, test, postmortem, and total.

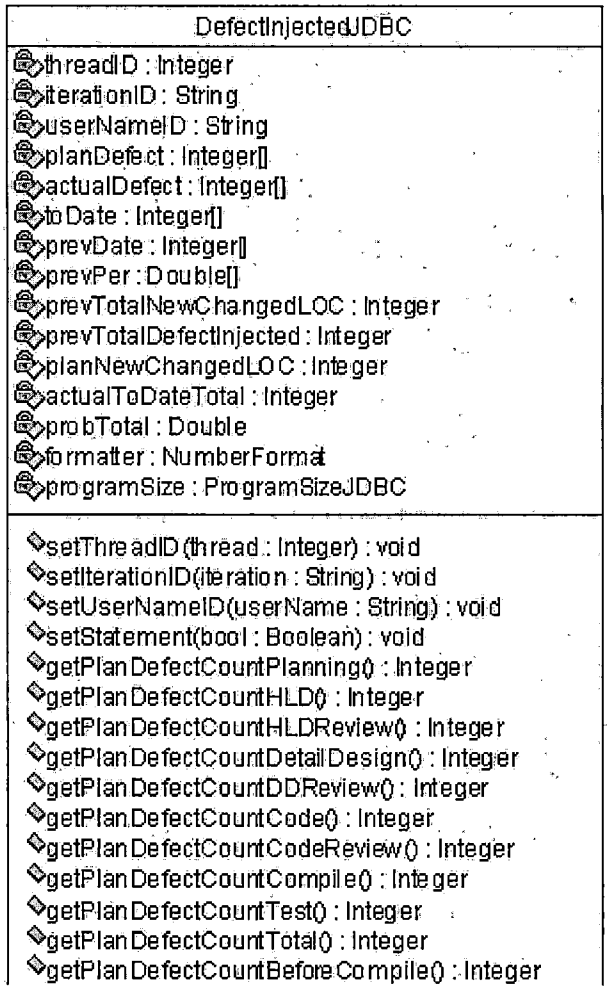


Figure 59. DefectInjectedJDBC Detailed Class Diagram

getActualDefectCountPlanning() : Integer
getActualDefectCountHLD() : Integer
getActualDefectCountHLDReview() : Integer
getActualDefectCountDetailDesign() : Integer
getActualDefectCountDDReview() : Integer
getActualDefectCountCode() : Integer
getActualDefectCountCodeReview() : Integer
getActualDefectCountCompile() : Integer
getActualDefectCountTest() : Integer
getActualDefectCountTotal() : Integer
getActualDefectCountBeforeCompile() : Integer
getToDateDefectCountPlanning() : Integer
getToDateDefectCountHLD() : Integer
getToDateDefectCountHLDReview() : Integer
getToDateDefectCountDetailDesign() : Integer
getToDateDefectCountDDReview() : Integer
getToDateDefectCountCode() : Integer
getToDateDefectCountCodeReview() : Integer
getToDateDefectCountCompile() : Integer
getToDateDefectCountTest() : Integer
getToDateDefectCountTotal() : Integer
getToDatePerDefectCountPlanning() : String
getToDatePerDefectCountHLD() : String
getToDatePerDefectCountHLDReview() : String
getToDatePerDefectCountDetailDesign() : String
getToDatePerDefectCountDDReview() : String
getToDatePerDefectCountCode() : String
getToDatePerDefectCountCodeReview() : String
getToDatePerDefectCountCompile() : String
getToDatePerDefectCountTest() : String
getToDatePerDefectCountTotal() : String
getToDatePerDefectCountBeforeCompile() : String

Figure 59. (Continued)

2.2.7 DefectRemovedJDBC Class

The purpose of DefectRemovedJDBC class is to access and update the PSP database for use by the DefectRemoved class. It accesses and updates the number of defects removed in plan, the number of defects removed in actual, the number of defects removed to date, and the to date defects removed percentage. It includes attributes:

planning, high level design, high level design review, detail design, detail design review, code, code review, compile, test, postmortem, and total.

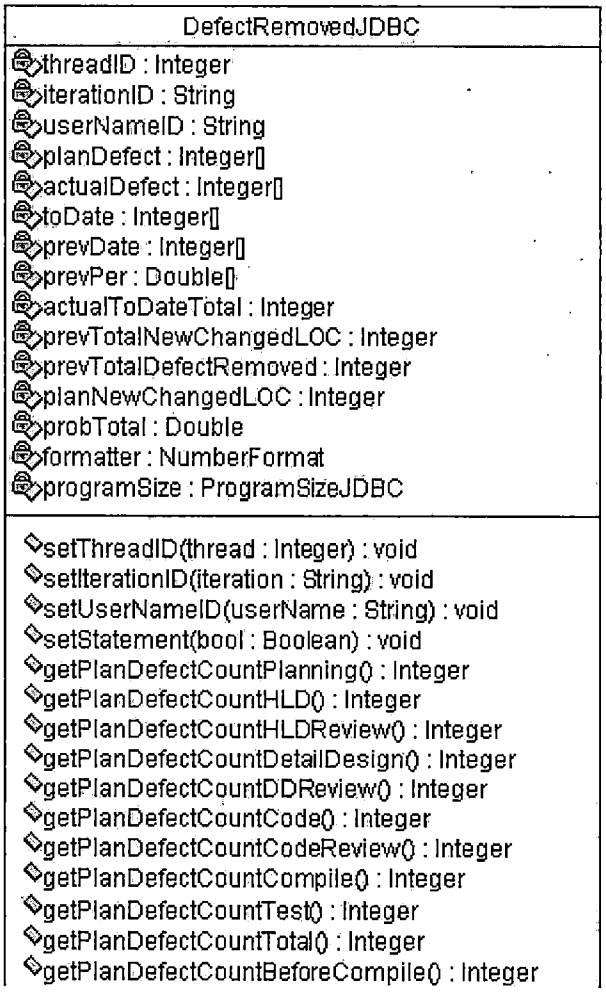


Figure 60. DefectRemovedJDBC Detailed Class Diagram

```
getActualDefectCountPlanning() : Integer
getActualDefectCountHLD() : Integer
getActualDefectCountHLDReview() : Integer
getActualDefectCountDetailDesign() : Integer
getActualDefectCountDDReview() : Integer
getActualDefectCountCode() : Integer
getActualDefectCountCodeReview() : Integer
getActualDefectCountCompile() : Integer
getActualDefectCountTest() : Integer
getActualDefectCountTotal() : Integer
getActualDefectCountBeforeCompile() : Integer
getToDateDefectCountPlanning() : Integer
getToDateDefectCountHLD() : Integer
getToDateDefectCountHLDReview() : Integer
getToDateDefectCountDetailDesign() : Integer
getToDateDefectCountDDReview() : Integer
getToDateDefectCountCode() : Integer
getToDateDefectCountCodeReview() : Integer
getToDateDefectCountCompile() : Integer
getToDateDefectCountTest() : Integer
getToDateDefectCountTotal() : Integer
getToDatePerDefectCountPlanning() : String
getToDatePerDefectCountHLD() : String
getToDatePerDefectCountHLDReview() : String
getToDatePerDefectCountDetailDesign() : String
getToDatePerDefectCountDDReview() : String
getToDatePerDefectCountCode() : String
getToDatePerDefectCountCodeReview() : String
getToDatePerDefectCountCompile() : String
getToDatePerDefectCountTest() : String
getToDatePerDefectCountTotal() : String
getToDatePerDefectCountBeforeCompile() : String
```

Figure 60. (Continued)

2.2.8 SizeEstimatingJDBC Class

The purpose of SizeEstimatingJDBC class is to access and update the PSP database for use by the SizeEstimating class. It accesses and updates base program, objects, and reused programs. Also, it performs the calculation of projected LOC, regression parameter, estimated new and

changed LOC, estimated total LOC, estimated new reused LOC, estimated total development time, prediction range, upper prediction interval, and lower prediction interval.

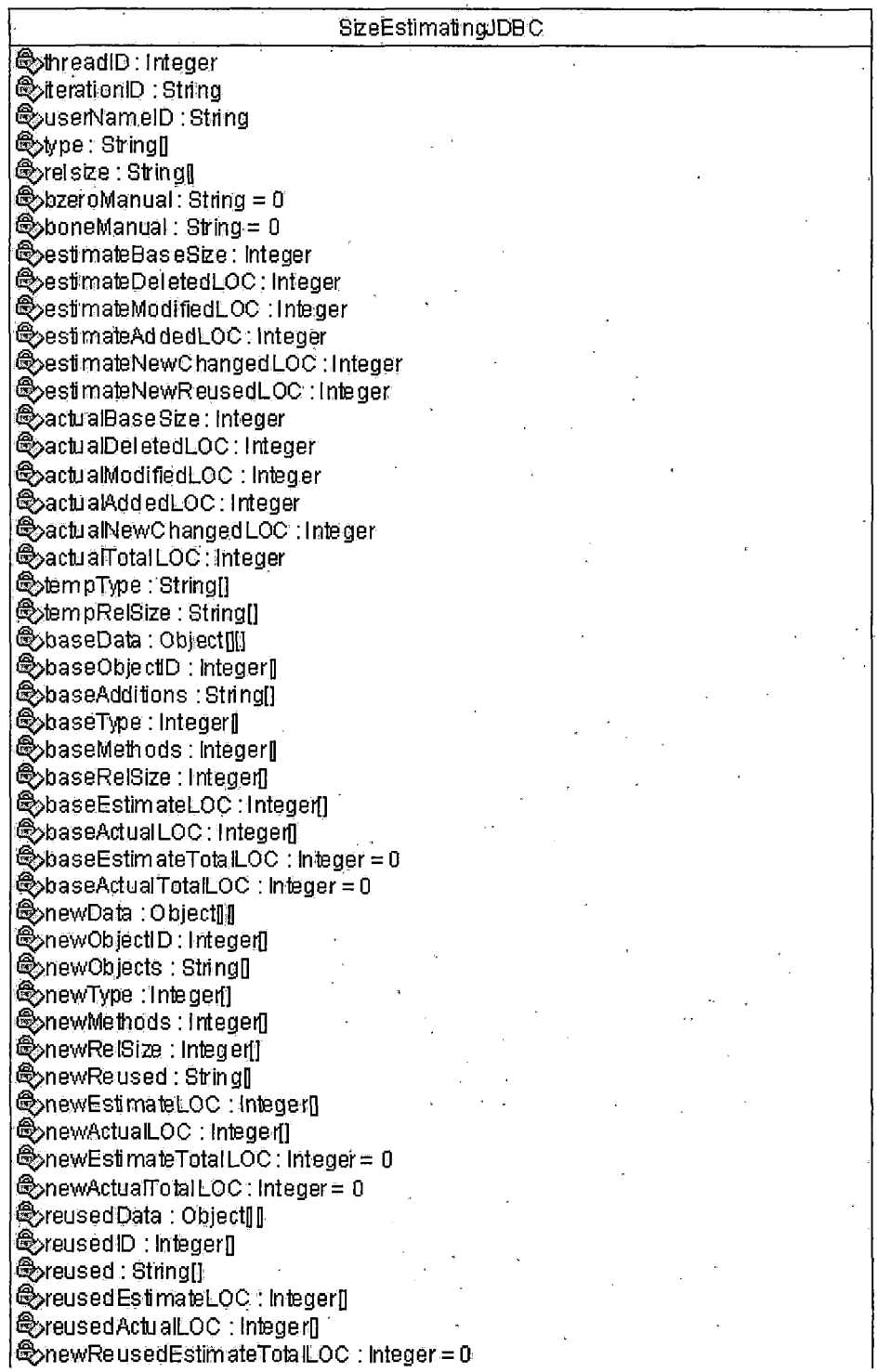


Figure 61. SizeEstimatingJDBC Detailed Class Diagram

newReusedActualTotalLOC : Integer = 0
reusedEstimateTotalLOC : Integer = 0
reusedActualTotalLOC : Integer = 0
projectedLOC : Integer
regressionZeroSize : Double
regressionZeroTime : Double
regressionOneSize : Double
regressionOneTime : Double
doubleEstNewChangedLOC : Double
estimateTotalLOC : Double
estimatePlanningTime : Integer
estimateHLDTime : Integer
estimateHLDReviewTime : Integer
estimateDetailDesignTime : Integer
estimateDDReviewTime : Integer
estimateCodeTime : Integer
estimateCodeReviewTime : Integer
estimateCompileTime : Integer
estimateTestTime : Integer
estimatePMTime : Integer
estimateTotalTime : Double
rangeSize : Double
rangeTime : Double
upiSize : Double
upiTime : Double
lpiSize : Double
lpiTime : Double
piPercentSize : Double
piPercentTime : Double
baseCount : Integer = 0
newCount : Integer = 0
reusedCount : Integer = 0
estimateLOCNewChanged : Vector
actualLOCNewChanged : Vector
actualTimeTotal : Vector
tDistribution70 : Double[]
tDistribution90 : Double

setThreadID(thread : Integer) : void
setIterationID(iteration : String) : void
setUserNameID(userName : String) : void
setStatement(bool : Boolean) : void
getEstimateBaseSize() : Integer
getActualBaseSize() : Integer
getEstimateDeletedLOC() : Integer
getActualDeletedLOC() : Integer
getEstimateModifiedLOC() : Integer
getActualModifiedLOC() : Integer
getBaseData() : Object[][]
getBaseEstimatedTotalLOC() : Integer

Figure 61. (Continued)

getBaseActualTotalLOC() : Integer
getNewData() : Object[][]
getNewEstimatedTotalLOC() : Integer
getNewActualTotalLOC() : Integer
getReusedData() : Object[][]
getReusedEstimatedTotalLOC() : Integer
getReusedActualTotalLOC() : Integer
getProjectedLOC() : Integer
getLOCNewChanged() : void
getActualTimeTotal() : void
computeAvg(data : Vector) : Double
computeBOne(xData : Vector, yData : Vector) : Double
computeBZero(xData : Vector, yData : Vector) : Double
computeStdDev(xData : Vector, yData : Vector) : Double
computeRange(xData : Vector, yData : Vector, t : Double, estTerm : Integer) : Double
getBzeroManual() : String
getBoneManual() : String
getSizeRegParaBZero() : Double
getSizeRegParaBOne() : Double
getTimeRegParaBZero() : Double
getTimeRegParaBOne() : Double
getEstimatedNewChangedLOC() : Double
getEstimatedTotalLOC() : Double
getEstimatedNewReusedLOC() : Double
getEstimatedTotalTime() : Double
getSizePredRange() : Double
getTimePredRange() : Double
getSizeUpperPredInt() : Double
getSizeLowerPredInt() : Double
getTimeUpperPredInt() : Double
getTimeLowerPredInt() : Double
setLOC(sizePhase : String, LOC : Integer) : void
setEstimateBaseSize(LOC : Integer) : void
setEstimateDeletedLOC(LOC : Integer) : void
setEstimateModifiedLOC(LOC : Integer) : void
setActualBaseLOC(LOC : Integer) : void
setActualDeletedLOC(LOC : Integer) : void
setActualModifiedLOC(LOC : Integer) : void
setBZeroManual(manual : String) : void
setBOneManual(manual : String) : void
setSizeRegParaBZero(bzero : Double) : void
setSizeRegParaBOne(bone : Double) : void
setTimeRegParaBZero(bzero : Double) : void
setTimeRegParaBOne(bone : Double) : void
setBaseData(data : Object[][]) : void
setNewData(data : Object[][]) : void
setReusedData(data : Double[][]) : void
getEstLOC() : Vector
getActLOC() : Vector
getActTime() : Vector
convType(temp : String[]) : Integer[]
convRelSize(temp : String[]) : Integer[]

Figure 61. (Continued)

2.2.9 TimeRecordingJDBC Class

The purpose of TimeRecordingJDBC class is to access and update the PSP database for use by the TimeRecording class.

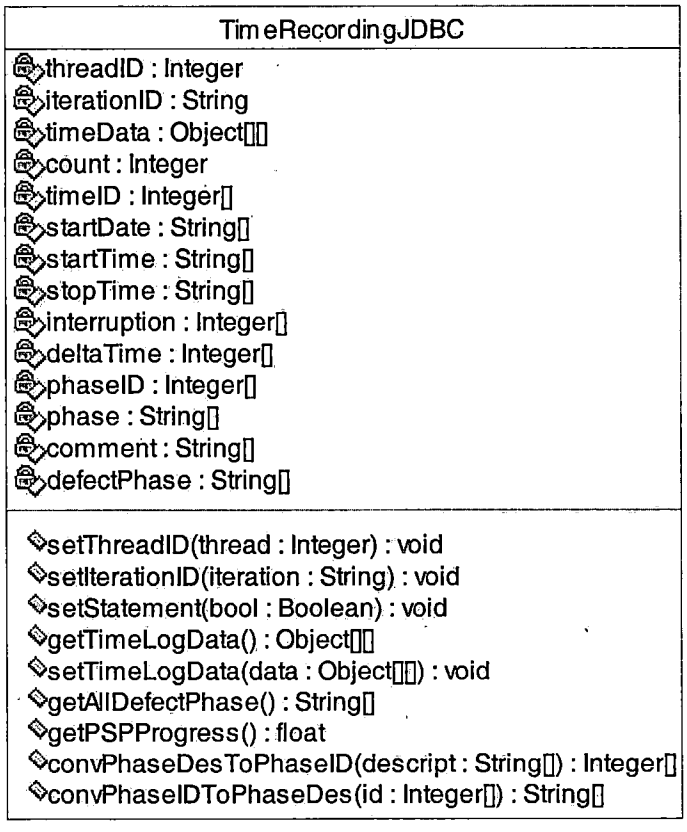


Figure 62. TimeRecordingJDBC Detailed Class Diagram

2.2.10 PIPProblemJDBC Class

The purpose of PIPProblemJDBC class is to access and update the PSP database for use by the ProcessImprovement

class. It accesses and updates the problem description in the Process Improvement Proposal form.

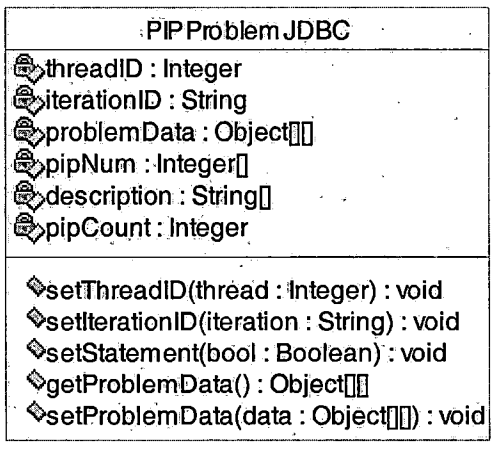


Figure 63. PIPProblemJDBC Detailed Class Diagram

2.2.11 PIPProposalJDBC Class

The purpose of PIPProposalJDBC class is to access and update the PSP database for use by the ProcessImprovement class. It accesses and updates the proposal description in the Process Improvement Proposal form.

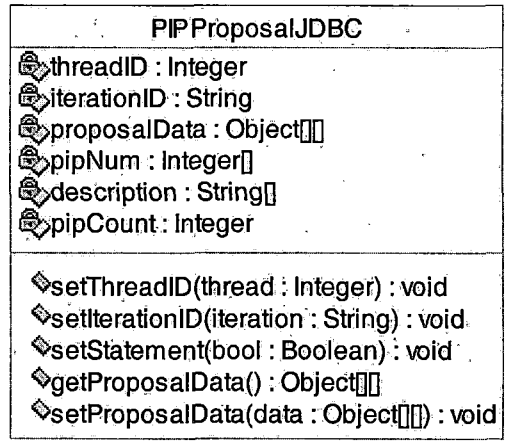


Figure 64. PIPProposalJDBC Detailed Class Diagram

2.3 Database Design

PSP database uses Oracle 8 as its database management system (DBMS), which will store the whole data needed for HTML forms and PSP applets. PSP database accepts the connection from the JDBC class, which executes SQL statements that the PSP database needs.

The following ER diagram shows the relationship of each table required by the PSP Scriber.

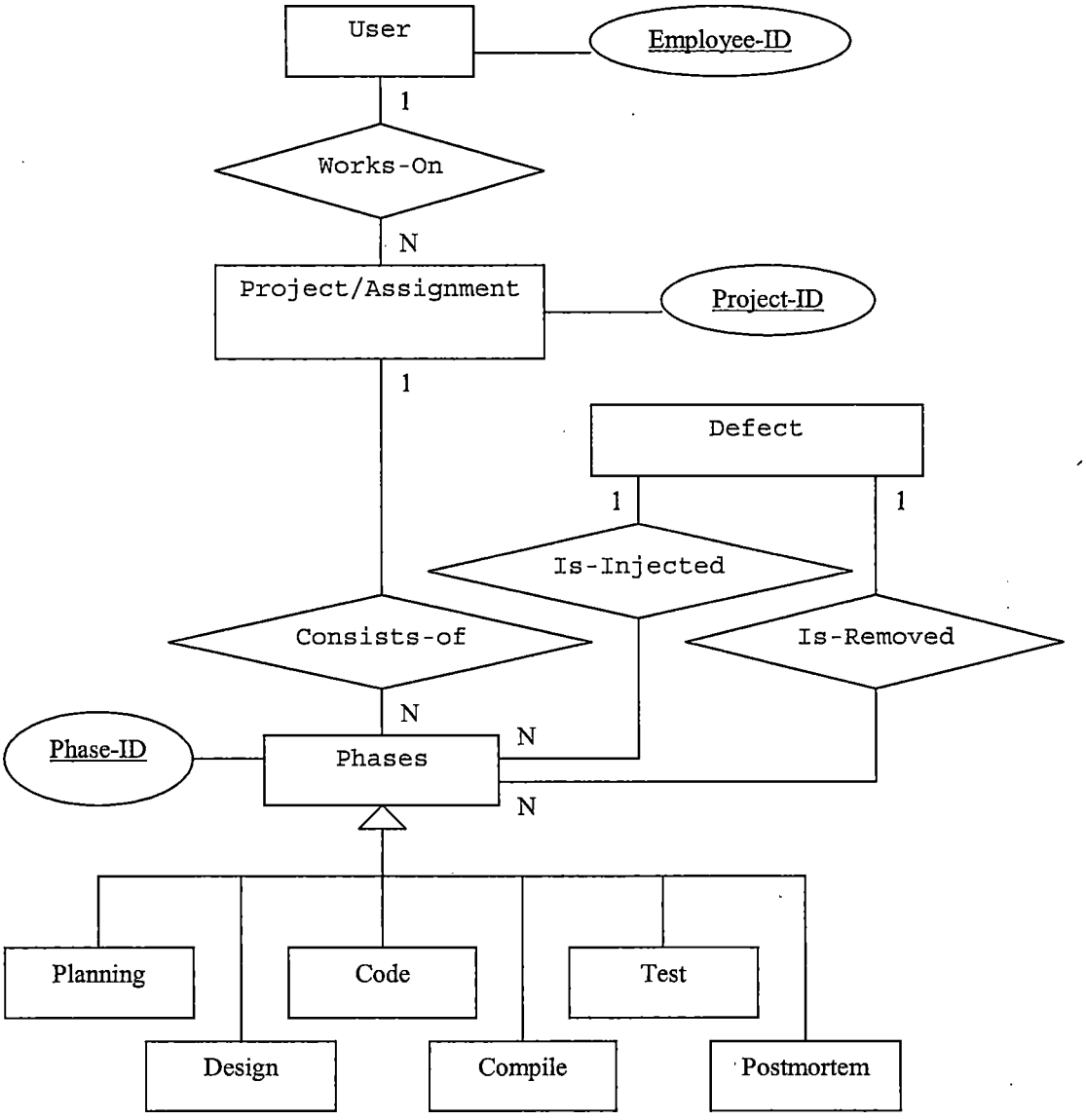


Figure 65. Database Entity Relationship Diagram

The following sub-section shows the detailed design of each database table.

2.3.1 Personnel Table

The personnel table provides information. The Purpose of Personnel table is to record the detail information for the users, which are users' first name, last name, employee ID, password, phone number, E-mail address, and password phrase.

Table 1. The Properties of Personnel Table

Field Name	Description
first_Name	User's first name.
last_Name	User's last name.
employee_ID	User's ID, given by administrator.
password	User's password for logging in.
phone1	User's primary phone number.
phone2	User's second phone number.
Email	User's e-mail address.
Pass Phrase	User's password phrase.

2.3.2 Psp_Project Table

The psp_project table provides information about a project undertaken by a user including project ID, project name, employee ID, starting date, and so on. Also, this table provides lines of code, time spending in each phase, number of defects injected in each phase, and number of defects removed in each phase for that particular project for the Project Plan Summary and Size Estimating Template.

Table 2. The Properties of Psp_Project Table

Field Name	Description
psp_ ID	PSP project ID for distinguishing every project.
psp_project_name	PSP project name.
starting_date	Starting date for this particular PSP project, e.g., 2001-11-20.
psp_progress	PSP progress level, e.g., 0, 0.1, 1.0, 1.1, 2.0, 2.1 or 3.0.
program_num	Program number, e.g., 1A, 2A.
supervisor	User's supervisor.
employee_ID	Employee ID, given by administrator
language	Programming language user used to write the program, e.g., C++, Java.
notes	Notes for Process improvement proposal.
p_loc_base	Planned base line of code.
p_loc_delete	Planned delete line of code.
p_loc_modified	Planned modified line of code.
p_loc_added	Planned added line of code.
p_loc_reused	Planned reused line of code.
p_loc_new_changed	Planned new and changed line of code.
p_loc_total	Planned total line of code.
p_loc_new_reused	Planned new reused line of code.
a_loc_base	Actual base line of code.
a_loc_delete	Actual delete line of code.
a_loc_modified	Actual modified line of code.
a_loc_added	Actual added line of code.
a_loc_reused	Actual reused line of code.
a_loc_new_changed	Actual new and changed line of code.
a_loc_total	Actual total line of code.
a_loc_new_reused	Actual new reused line of code.
p_time_planning	Estimate time, in minutes, for planning.
p_time_HLD	Estimate time for high-level design.
p_time_HLD_review	Estimate time for high-level design review.

p_time_detail_design	Estimate time for detail design.
p_time_DD_review	Estimate time for detail design review.
p_time_code	Estimate time for coding.
p_time_code_review	Estimate time for code review.
p_time_compile	Estimate time for compile.
P_time_test	Estimate time for test.
P_time_postmortem	Estimate time for postmortem.
A_time_planning	Actual time, in minutes, for planning.
A_time_HLD	Actual time for high-level design.
A_time_HLD_review	Actual time for high-level design review.
A_time_detail_design	Actual time for detail design.
A_time_DD_review	Actual time for detail design review.
A_time_code	Actual time for coding.
A_time_code_review	Actual time for code review.
A_time_compile	Actual time for compile.
A_time_test	Actual time for test.
A_time_postmortem	Actual time for postmortem.
P_defect_inj_planning	Estimate defect injected on planning.
P_defect_inj_HLD	Estimate defect injected on high-level design.
P_defect_inj_HLD_review	Estimate defect injected on high-level design review.
P_defect_inj_detailed_design	Estimate defect injected on detailed design.
P_defect_inj_DD_review	Estimate defect injected on detailed design review.
P_defect_inj_code	Estimate defect injected on coding.
P_defect_inj_code_review	Estimate defect injected on code review.
P_defect_inj_compile	Estimate defect injected on compile.
P_defect_inj_test	Estimate defect injected on test.
A_defect_inj_planning	Actual defect injected on planning.
a_defect_inj_HLD	Actual defect injected on high-

	level design.
a_defect_inj_HLD_review	Actual defect injected on high-level design review.
a_defect_inj_detailed_design	Actual defect injected on detailed design.
a_defect_inj_DD_review	Actual defect injected on detail design review.
a_defect_inj_code	Actual defect injected on coding.
a_defect_inj_code_review	Actual defect injected on code review.
a_defect_inj_compile	Actual defect injected on compile.
a_defect_rem_test	Actual defect injected on test.
p_defect_rem_planning	Estimate defect removed on planning.
p_defect_rem_HLD	Estimate defect removed on high-level design.
p_defect_rem_HLD_review	Estimate defect removed on high-level design review.
p_defect_rem_detailed_design	Estimate defect removed on detailed design.
p_defect_rem_DD_review	Estimate defect removed on detailed design review.
p_defect_rem_code	Estimate defect removed on coding.
p_defect_rem_code_review	Estimate defect removed on code review.
p_defect_rem_compile	Estimate defect removed on compile.
p_defect_rem_test	Estimate defect removed on test.
A_defect_rem_planning	Actual defect removed on planning.
a_defect_rem_HLD	Actual defect removed on high-level design.
a_defect_rem_HLD_review	Actual defect removed on high-level design review.
a_defect_rem_detailed_design	Actual defect removed on detailed design.
a_defect_rem_DD_review.	Actual defect removed on detailed design review.
a_defect_rem_code	Actual defect removed on coding.
a_defect_rem_code_revie	Actual defect removed on code

w	review.
a_defect_rem_compile	Actual defect removed on compile.
a_defect_rem_test	Actual defect removed on test.

2.3.3 Time_Log Table

The time_log table record the time spent in each project phase. It contains start time, stop time, interruption time, phase, and comments for each time logged.

Table 3. The Properties of Time_Log Table

Field Name	Description
psp_ID	PSP project ID.
time_ID	Sequential number for time recording log.
start_time	Date and time when user starts working on a task.
stop_time	Date and time when user stops working on a task.
interruption_time	Interruption time, in minutes, that was not spending on the task.
phase_ID	Phase ID for phase description.
Comments	Comments for the task.

2.3.4 Defect_Log Table

The defect_log table holds the data on each defect as you find and correct it. It contains occur date, defect number, defect type, inject phase, remove phase, fix time, fix defect, and description.

Table 4. The Properties of Defect_Log Table

Field Name	Description
psp_ID	PSP project ID.
occurt_date	The date when defect was found, e.g., 2001-11-20.
defect_num	Defect sequential number starting with 1.
defect_ID	Defect type ID from defect type standard.
inject_ID	The phase during which this defect was injected.
remove_ID	The phase during which this defect was removed.
fix_time	The time, in minutes, user took to fix the defect.
fix_defect	Fix another defect number.
description	Succinct description of the defect.

2.3.5 Task_Plan Table

The task_plan table provides information about items used in planning a task. It contains task number, task name, plan hours, planning date, and actual date.

Table 5. The Properties of Task_Plan Table

Field Name	Description
psp_ID	PSP project ID.
task_number	Sequential number for task planning template.
task_name	Task phase name.
hours	Planning hours for each task.
plan_date	Plan date on the Task Planning Template, e.g., 2001-11-20.
actual_date	Completion date, e.g., 2001-11-20.

2.3.6 Schedule_Plan Table

The schedule_plan table provides information about items used in scheduling a project. It contains week number, date, planning hours, and actual hours.

Table 6. The Properties of Schedule_Plann Table

Field Name	Description
psp_ID	PSP project name.
week_number	Week sequential number starting with 1.
date_monday	Calendar date for each week
plan_hour	Direct project hours user expect to spend each week.
actual_hour	Actual direct hours.

2.3.7 Pip_Proposal Table

The pip_proposal table provides description of the Process Improvement Proposal encountered in the problem and for improvement ideas. It contains proposal number and description for each proposal.

Table 7. The Properties of Pip_Proposal Table

Field Name	Description
psp_ID	PSP project ID.
proposal_number	Proposal sequential number starting with 1.
description	Describe user's proposed process improvement.

2.3.8 Pip_Problem Table

The pip_problem table provides description in the Process Improvement Proposal to provide an orderly record of user's process improvement ideas for use in later process improvement. It contains problem number and description for each problem.

Table 8. The Properties of Pip_Problem Table

Field Name	Description
psp_ID	PSP project ID.
pip_number	Problem sequential number starting with 1.
description	Describe the problem.

2.3.9 Defect_Type Table

The defect_type table provides defect type description for the Defect Recording Log.

Table 9. The Properties of Defect_Type Table

Field Name	Description
defect_ID	Defect type number.
description	Defect type name.

2.3.10 Defect_Phase Table

The defect_phase table provides the phase description for the Project Plan Summary, the Time Recording log, the Defect Recording Log, and the Task Planning Template.

Table 10. The Properties of Defect_Phase Table

Field Name	Description
phase_ID	Defect phase number.
description	Defect phase name.

2.3.11 Test_Report Table

The test_report table provides information for the Test Report Template to maintain a record of the tests run and results obtained, and to be sufficiently complete so you can later repeat the same tests and get the same results. It contains test ID, test name, test objective, test description, test conditions, expected results, and actual results.

Table 11. The Properties of Test_Report Table

Field Name	Description
psp_ID	PSP project ID.
test_ID	Test sequential number starting with 1.
test_name	Uniquely identify each test execution for each program.
test_object	Briefly describe the test objective.
description	Describe each test data and procedures.
conditions	List any special configuration, timing, fix, or other conditions of the test.
exp_result	List the results that the test should produce if it runs properly.
act_result	List the result that were actually produced.

2.3.12 Projected_Log Table

The projected_log provides information for the base additions and new objects in the Size Estimating Template to hold the actual and estimate data. It contains object name, type, numbers of method, relative size, estimate lines of code, and actual lines of code.

Table 12. The Properties of Projected_Log Table

Field Name	Description
psp_ID	PSP project ID.
object_name	Identify the functions to be added, or assign a name to planned object.
kind	Kind of object, e.g., base addition or new object.
type	Object type, e.g., logic, I/O, calculation, text, data, or set-up.
methods	Number of methods the object contains.
rel_size	The relative size of the object, e.g., very small, small, medium, large, very large.
estimate_loc	Estimated line of code for the object.
actual_loc	Actual line of code for the object.

2.3.13 Reused_Programs Table

The reused_programs table provides information for the reused programs in the Size Estimating Template to hold the actual and estimate data. It contains name, estimate lines of code, and actual lines of code.

Table 13. The Properties of Reused_Programs Table

Field Name	Description
psp_ID	PSP project ID.
name	Name of unmodified reused object.
estimate_loc	Unmodified reused object estimate line of code.
actual_loc	Unmodified reused object actual line of code.

2.4 Testing

PSP Scriber was tested by students registered in Fall 2001 CSCI655 Software Engineering class in the department of Computer Science, California Stat University, San Bernardino. The detail testing is described in Appendix A: Detail Testing Results. The following sub-section describer the error handling of wrong data.

2.4.1 Error Handling in the PSP0 Project Plan Summary

If the user enters improper date in the PSP0 Project Plan Summary for example, the PSP Scriber will cache the error and display an error dialog box.

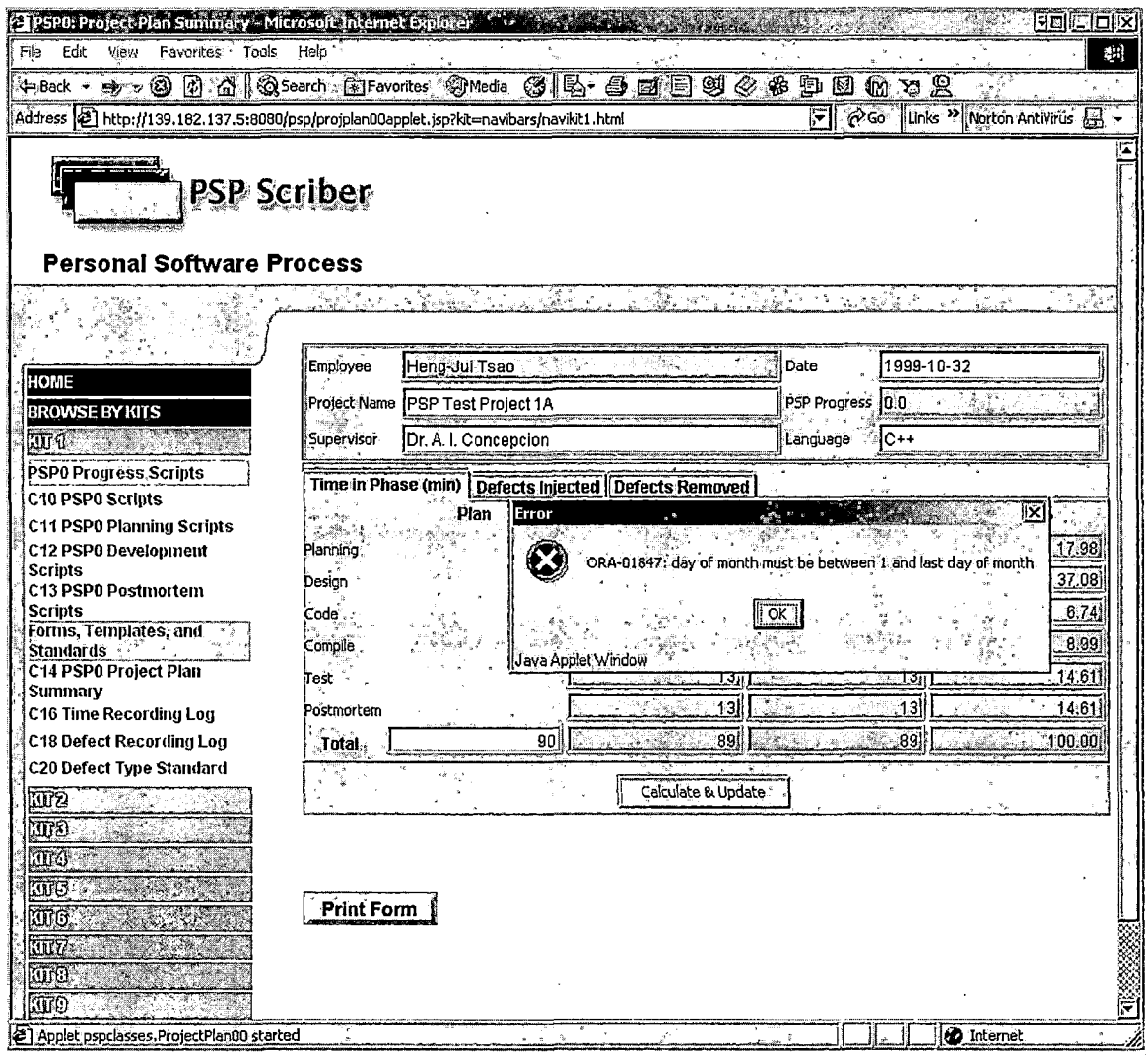


Figure 66. Screen Shot of Error Handling in the PSP0 Project Plan Summary

2.4.2 Error Handling in the Time Recording Log

If the user enters improper minutes in the Time Recording Log for example, the PSP Scriber will cache the error and display an error dialog box.

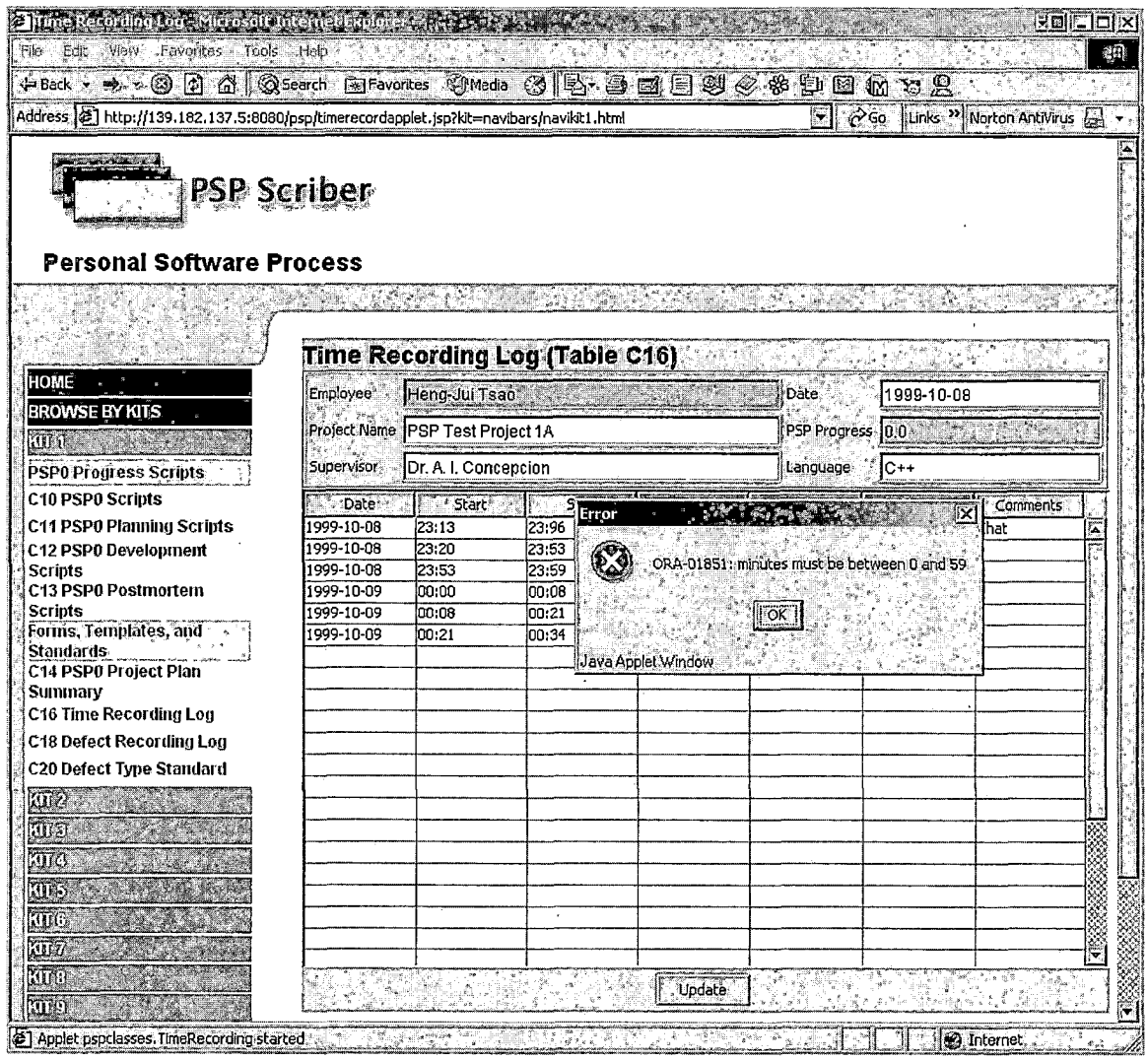


Figure 67. Screen Shot of Error Handling in the Time Recording Log

2.4.3 Error Handling in the Defect Recording Log

If the user enters improper month in the Defect Recording Log for example, the PSP Scriber will cache the error and display an error dialog box.

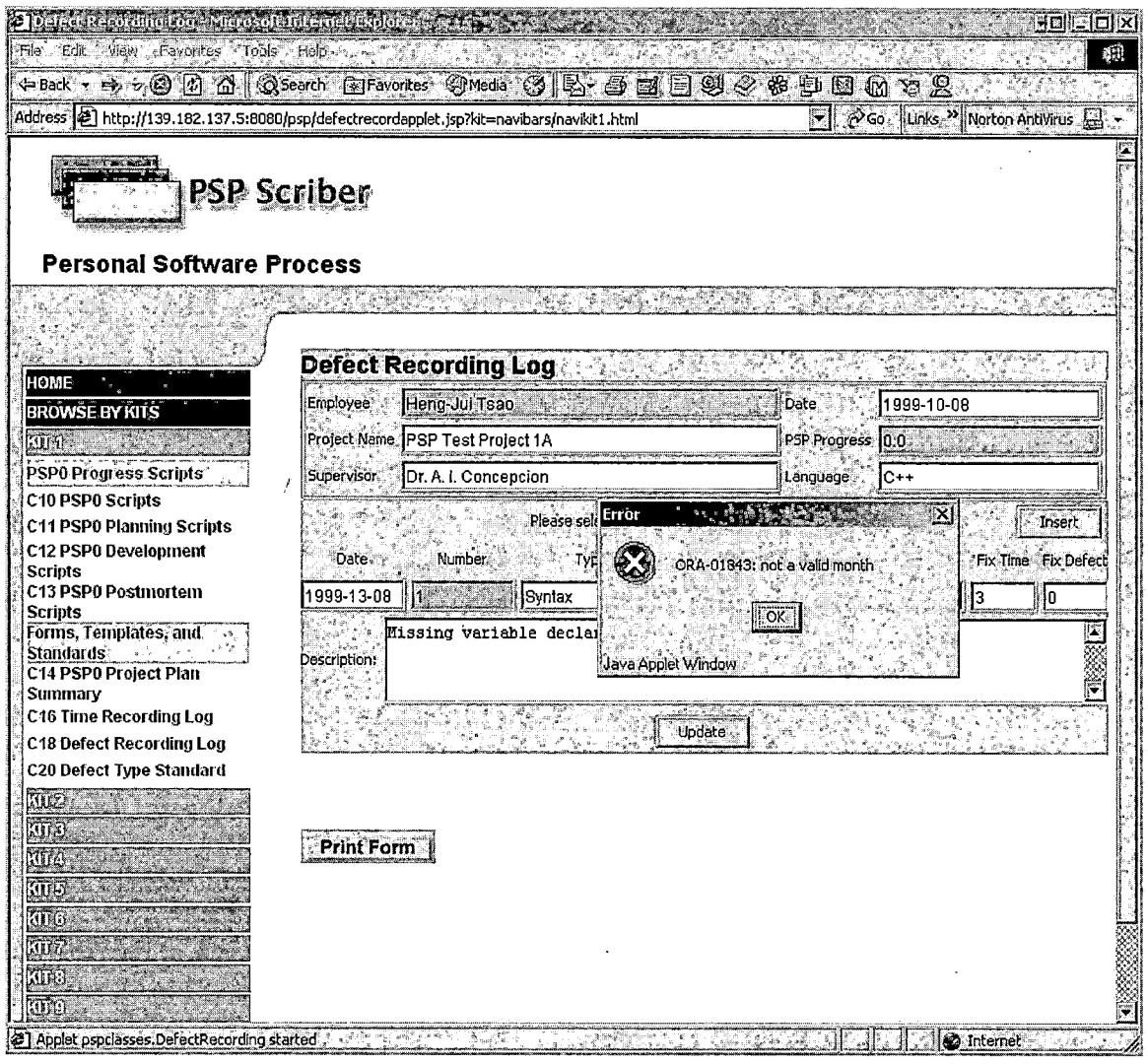


Figure 68. Screen Shot of Error Handling in the Defect Recording Log

CHAPTER THREE

MAINTENANCE MANUAL

3.1 Source Code

In the PSP Scriber project, there are thirty-eight .java files, forty-four .jsp files, forty-nine .html files, and twelve .sql files. All source files are stored in the attached CD-ROM.

Here is a list of source files.

- The Java source files are shown in Table 14.

Table 14. Java Source Files

AuthorizationJDBC.java	DefectDetailJDBC.java
DefectInjected0.java	DefectInjected20.java
DefectInjectedJDBC.java	DefectRecording.java
DefectRemoved0.java	DefectRemoved20.java
DefectRemovedJDBC.java	JDBCDriver.java
MyTableModel.java	PIPProblem.java
PIPProposal.java	ProcessImprovement.java
ProgramSize10.java	ProgramSizeJDBC.java
ProjectInfo.java	ProjectInfoJDBC.java
ProjectPlan00.java	ProjectPlan01.java
ProjectPlan10.java	ProjectPlan11.java
PeojectPlan20.java	PSPAuthorization.java
SchedulePlan.java	SchedulePlanJDBC.java
SizeEstimating.java	SizeEstimatingJDBC.java
TaskPlan.java	TaskPlanJDBC.java
TestReport.java	TestReportJDBC.java
TimeInPhase01.java	TimeInPhase20.java
TimeInPhaseJDBC.java	TimeRecording.java
TimeRecordingJDBC.java	

- The JSP source files are shown in Table 15.

Table 15. JSP Source Files

authorization.jsp	defectrecordapplet.jsp
index.jsp	kit01main.jsp
kit02main.jsp	kit03main.jsp
kit04main.jsp	kit05main.jsp
kit06main.jsp	kit07main.jsp
kit08main.jsp	kit09main.jsp
kit10main.jsp	kit11main.jsp
pdefectrecording.jsp	pprocessimprovement.jsp
pprojectplan00.jsp	pprojectplan01.jsp
pprojectplan10.jsp	pprojectplan11.jsp
pprojectplan20.jsp	pprojectplan21.jsp
pprojectplan30.jsp	processimprovementapplet.jsp
projectplan00applet.jsp	projectplan01applet.jsp
projectplan10applet.jsp	projectplan11applet.jsp
projectplan20applet.jsp	psizeestimating.jsp
pspmain.jsp	pspscript.jsp
ptestreport.jsp	ptimerecording.jsp
register.jsp	scheduleplanapplet.jsp
sizeestimatingapplet.jsp	taskplanapplet.jsp
testreportapplet.jsp	timerecordapplet.jsp

- The HTML source files are shown in Table 16.

Table 16. HTML Source Files

kit1.html	kit10.html
kit11.html	kit2.html
kit3.html	kit4.html
kit5.html	kit6.html
kit7.html	kit8.html
kit9.html	indextop.html
login.html	navigation.html
navi1.html	navi10.html
navi11.html	navi2.html
navi3.html	navi4.html
navi5.html	navi6.html
navi7.html	navi8.html
navi9.html	projecttop.html
scr_c10.html	scr_c10.html
scr_c11.html	scr_c12.html
scr_c13.html	scr_c15.html

scr_c17.html	scr_c19.html
scr_c20.html	scr_c21.html
scr_c22.html	scr_c23.html
scr_c24.html	scr_c26.html
scr_c28.html	scr_c30.html
scr_c31.html	scr_c32.html
scr_c33.html	scr_c35.html
scr_c36.html	scr_c38.html
scr_c40.html	welcome.html

- The SQL source files are shown in Table 17.

Table 17. SQL Source Files

defect_log.sql	defect_phase.sql
defect_type.sql	pip_problem.sql
pip_proposal.sql	projected_log.sql
pip_project.sql	reused_programs.sql
schedule_plan.sql	task_plan.sql
test_report.sql	time_log.sql

3.2 Re-compile

This section shows how to re-compile the PSP Scriber whenever any Java source file has been changed. The following sub-section will describe how to use Borland Jbuilder Java compiler to re-compile Java classes for the PSP Scriber.

1. Put all .java source files into one directory.
2. Open Jbuilder, choose File | New.
3. Double click Project icon to create a new project.

4. In the project name, type "pspClasses" in the text field.
5. Click Finish when done.
6. Choose Project | Add files / Packages.
7. In the dialog window, explore your .java source files directory.
8. Select all files in this directory and click OK.
9. Choose Wizards | Archive Builder.
10. In the Archive type, select applet in the combo box and click next.
11. For the Name, type "PSPAppletsTomcat".
12. Click the explore button after Files text field.
13. In the File name field, type "pspApplets.jar".
14. Click OK to close this dialog.
15. Click Next.
16. Click Add Class button, locate oracle.jdbc.driver.OracleDriver class, and click OK.
17. Click Next.
18. Choose Oracle Thin Driver, under Library settings, select "Always include all classes and resource".

19. Click Finish to end this progress.

20. Choose Wizards | Archive Builder.

21. Click next without any change.

22. For the Name, type "PSPClassesTomcat".

23. Click the explore button after Files text field.

24. In the File name field, type "pspClasses.jar".

25. Click OK to close this dialog.

26. Click Next.

27. Click Next without any change.

28. Choose Oracle Thin Driver, under Library settings, select "Always include all classes and resource".

29. Click Finish to end this progress.

30. Choose Project | Rebuild Project.

Two archive file "pspApplets.jar" and "pspClasses.jar" are created.

3.3 Installation Process

1. Put the files in Table 18 in the root directory of the Web server.

Table 18. Files in Root Directory

authorization.jsp	defectrecordapplet.jsp
index.jsp	kit01main.jsp
kit02main.jsp	kit03main.jsp

kit04main.jsp	kit05main.jsp
kit06main.jsp	kit07main.jsp
kit08main.jsp	kit09main.jsp
kit10main.jsp	kit11main.jsp
processimprovementapplet.jsp	projectplan00applet.jsp
projectplan01applet.jsp	projectplan10applet.jsp
projectplan11applet.jsp	projectplan20applet.jsp
pspmain.jsp	pspscript.jsp
register.jsp	scheduleplanapplet.jsp
sizeestimatingapplet.jsp	taskplanapplet.jsp
testreportapplet.jsp	timerecordapplet.jsp
indextop.html	login.html
projecttop.html	welcome.html

2. Put the files in Table 19 in the sub-directory print/ of the root directory

Table 19. Files in Print/ Sub-Directory

pdefectrecording.jsp	pprocessimprovement.jsp
pprojectplan00.jsp	pprojectplan01.jsp
pprojectplan10.jsp	pprojectplan11.jsp
pprojectplan20.jsp	pprojectplan21.jsp
pprojectplan30.jsp	psizeestimating.jsp
ptestreport.jsp	ptimerecording.jsp

3. Put the files in Table 20 in the sub-directory navibars/ of the root directory

Table 20. Files in Navibars/ Sub-Directory

navigation.html	navi1.html
navi10.html	navi11.html
navi2.html	navi3.html
navi4.html	navi5.html
navi6.html	navi7.html
navi8.html	navi9.html

4. Put the files in Table 21 in the sub-directory kits/ of the root directory

Table 21. Files in Kits/ Sub-Directory

kit1.html	kit10.html
kit11.html	kit2.html
kit3.html	kit4.html
kit5.html	kit6.html
kit7.html	kit8.html
kit9.html	

5. Put the files in Table 22 in the sub-directory scripts/ of the root directory

Table 22. Files in Scripts/ Sub-Directory

scr_c10.html	scr_c10.html
scr_c11.html	scr_c12.html
scr_c13.html	scr_c15.html
scr_c17.html	scr_c19.html
scr_c20.html	scr_c21.html
scr_c22.html	scr_c23.html
scr_c24.html	scr_c26.html
scr_c28.html	scr_c30.html
scr_c31.html	scr_c32.html
scr_c33.html	scr_c35.html
scr_c36.html	scr_c38.html
scr_c40.html	

6. Put the pspApplets.jap in the root directory.
7. Put the pspClasses.jar in the lib/ directory of the Web server.
8. For Oracle database, run the files in Table 18 under sqlplus console.

Table 23. Files for PSP Database

defect_log.sql	defect_phase.sql
defect_type.sql	pip_problem.sql
pip_proposal.sql	projected_log.sql
pip_project.sql	reused_programs.sql
schedule_plan.sql	task_plan.sql
test_report.sql	time_log.sql

3.4 User Manual

The PSP Scriber project uses Java applets and JSP pages to implement the Personal Software Process. PSP Scriber will support all the forms that PSP needs.

3.4.1 Before Using PSP Scriber

The user must register with the PSP Scriber administrator in order to get login ID and password. Users can ask PSP Scriber administrator for log-ID and password.

3.4.2 Overall Description

The PSP Scriber basically has two parts. One is the navigation index. This index navigation bar enables users to navigate all the PSP Scriber pages by simply clicking any of them. The second part is for displaying forms and Java applets. The Java applet will be described in the following section.

3.4.3 Java Applets User Manuals

The Java applets are the main functions that PSP Scriber used to support Personal Software Process. They

performs data input, calculation, and data update. The user should take note of the following,

- User should start from kit 1. Even though the user has done any previous kits by hand, the user should enter the data form kit 1 through current kit. This is necessary to create a record for all your information. They will be used in the succeeding kits.
- Because Java applet is slow, the user may need to wait some time to load the applet. Then after the user has updated the data, the user "must" click the update button, before leaving this page. After the database transmission is finished, the applet will show a dialog box to inform the user that the database transmission is finished. Then the user can leave this page.
- Once a user inputs an improper data into the Java Applet, the applet will show an error message box. The user must check the data just inputted and click the update button again.
- The user must follow the date format of "YYYY-MM-DD", and the time format of "HH-MM".

The following sub-sections describe more details about each Java applets.

3.4.3.1 PSP0 Project Plan Summary. The following steps describe the usages of the PSP0 Project Plan Summary:

1. When the user loads this applet for the first time, it will show "User Name", "Date", and "PSP Progress".
2. Enter "Project Name".
3. Enter "Supervisor".
4. Enter "Language", for example, C++ or Java.
5. Update "Date" with the format of "YYYY-MM-DD". The user must enter the exact date in order to calculate "To Date" field in the future.
6. Enter "Plan Total" time in "Time in Phase" section.
7. Click "Calculate & Update" button for updating PSP database. If PSP Scriber does not show any error messages and displays a confirm dialog box, PSP Scriber has finished the database updating.
8. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields in the Project Plan Summary will be updated automatically after the user has inserted records on the "Time Recording Log" and the "Defect Recording Log".

3.4.3.2 Time Recording Log. In the Time Recording Log, user must enter data on the first column "Date", for each row if these rows have data on it. The following steps describe the usage of the Time Recording Log:

1. Enter date with the format of "YYYY-MM-DD".
2. Enter "Start" time with the format of "HH:MM".
3. Enter "Stop" time with the format of "HH:MM".
4. Enter "Interruption Time" as an integer., or leave it blank when the interruption time is zero.
5. The user does not need to enter "Delta Time", PSP Scriber will calculate this.
6. Select a phase (e.g. planning, design, code, compile, test, postmortem) in the "Phase" combo box.
7. Enter "Comments" as necessary.
8. Repeat steps 1 through 7 if the user needs to add more rows in the Time recording Log.

9. Click "Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then PSP Scriber has finished the database updating.
10. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

3.4.3.3 Defect Recording Log. In Defect Recording Log, PSP Scriber has setup the first defect for the user. The following steps describe the usage of the Defect Recording Log:

1. Update the "Date" text field in the format of "YYYY-MM-DD". PSP Scriber has set the date for current date when user inserts a new defect record. The user may change the date if needed.
2. Select a type in the "Type" combo box.
3. Select a phase in the "Inject" combo box.
4. Select a phase in the "Remove" combo box.
5. Enter "Fix Time" as an integer.
6. Enter "Fix Defect" as necessary.
7. Enter "Description" for this defect.

8. Click "Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, the PSP Scriber has finished the database updating.
9. If the user needs to insert a new defect record, he/she may click "Insert" button for inserting a new record. Repeat steps 1 to 8 for updating this defect record.
10. Click "Print Form" button if the user needs to print this form with HTML format. User can print the form later after he/she finishes this whole assignment kit.

User can select a particular defect number for updating defect record in the future.

3.4.3.4 PSP0.1 Project plan Summary. The following steps describe the usage of the PSP0.1 Project Plan Summary:

1. When the user loads this applet for the first time, it will show "User Name", "Date", and "PSP Progress".
2. Enter "Project Name".
3. Enter "Supervisor".

4. Enter "Language", for example, C++ or Java.
5. Update "Date" in the format of "YYYY-MM-DD". The user must enter the exact date in order to calculate "To Date" field in the future.
6. Enter plan "Total New & Change" LOC in "Program Size" section.
7. Enter actual "Base", "Deleted", "Modified", "Reused", "Total", and "Total New Reused" LOC with the actual LOC for the program the user wants to record.
8. Enter plan "Planning", "Design", "Code", "Compile", "Test", and "Postmortem" time.
9. Click "Calculate & Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then PSP Scriber has finished the database update. Some fields of this form will be calculated automatically at this point.
10. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields (e.g. defect injected or defect removed) in the PSP0.1 Project Plan Summary will be updated automatically after the user has inserted records in the "Time Recording Log" and the "Defect Recording Log".

3.4.3.5 Process Improvement Proposal. In the Process Improvement Proposal, "PIP Number" and "Proposal PIP Number" are two fields the user must enter. The user must enter sequential number starting from 1 for these two fields if he/she needs to enter data on "Description". The following steps describe the usage of the Process Improvement Proposal.

1. Enter "PIP Number" with integer.
2. Enter "Problem Description".
3. Repeat steps 1 to 2 if the user needs more records on "Problem Description".
4. Enter "Proposal PIP Number" with integer.
5. Enter "Proposal Description".
6. Repeat steps 4 to 5 if the user needs more records on "Proposal Description".
7. Click "Update" button for updating PSP database. If the PSP Scriber does not show any error

message and displays a confirm dialog box, then the PSP Scriber has finished the database update.

8. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

3.4.3.6 PSP1 Project Plan Summary. The following steps describe the usages of the PSP1 Project Plan Summary.

1. When the user loads this applet for the first time, it will show "User Name", "Date", and "PSP Progress".
2. Enter "Project Name".
3. Enter "Supervisor".
4. Enter "Language", for example, C++ or Java.
5. Update "Date" with the format of "YYYY-MM-DD". The user must enter the exact date in order to calculate "To Date" field in the future.
6. Enter plan "Base", "Deleted", and "Modified" LOC.
7. Enter actual "Base", "Deleted", "Modified", and "Total" LOC with the actual LOC for the program the user wants to record.

8. Enter plan "Planning", "Design", "Code", "Compile", "Test", and "Postmortem" time.
9. Click "Calculate & Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then the PSP Scriber has finished the database update. Some fields of this form will be calculated automatically at this point.
10. Click "Print Form" button if the user need to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields (e.g. defect injected or defect removed) in the PSP1 Project Plan Summary will be updated automatically after the user has inserted records in the "Time Recording Log" and the "Defect Recording Log".

3.4.3.7 Test Report Template. In the Test Report Template, PSP Scriber has generated the first record set for the user. If the user needs to insert a new record set, he/she can click "Insert" button for inserting a new record. The following steps describe the usages of the Test Report Template.

1. Enter "Test Name/Number".
2. Enter "Test Objective".
3. Enter "Test Description".
4. Enter "Test Conditions".
5. Enter "Expected Results".
6. Enter "Actual Results".
7. Click "Update" button for updating PSP database. If PSP Scriber does not show any error messages and displays a confirm dialog box, PSP Scriber has finished the database updating.
8. Click "Print Form" button if the user need to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.
9. Click "Insert" button if the user needs to insert a new record for the Test Report Proposal.

The user can select a particular test number for updating a test record in the future.

3.4.3.8 Size Estimating Template. The following steps describe the usage of the Size Estimating Template.

1. Enter "Estimate base size".
2. Enter "Actual base size".
3. Enter "Estimate LOC deleted".
4. Enter "Actual LOC deleted".
5. Enter "Estimate LOC modified".
6. Enter "Actual LOC modified".
7. Enter name of "Base Addition".
8. Select type of "Base Addition".
9. Enter number of method for "Base Additions".
10. Select Relative Size of "Base Additions".
11. Enter Estimate LOC for "Base Additions".
12. Enter Actual LOC for "Base Additions".
13. Repeat steps 7 to 12 if the user has more "Base Additions".
14. Enter name of "New Object".
15. Select type of "New Object".
16. Enter number of method for "New Object".
17. Select Relative Size of "New Object".

18. Select "New Reused" if this "New Object" is new and reused.
19. Enter Estimate LOC for "New Object".
20. Enter Actual LOC for "New Object".
21. Repeat steps 14 to 19 if the user has more "New Object".
22. Enter name of "Reused Programs".
23. Enter estimate LOC for "Reused Programs".
24. Enter actual LOC for "Reused Programs".
25. Repeat steps 21 to 23 if the user has more "Reused Programs".
26. If the user needs to manually input a regression parameter, select appropriate check box on the left side of estimate LOC. The user can input regression parameter manually.
27. Click "Update" button for updating PSP database. If PSP Scriber does not show any error messages and displays a confirm dialog box, PSP Scriber has finished the database updating.
28. Click "Print Form" button if the user needs to print this form with HTML format. The user can

print the form later after he/she finishes this whole assignment kit.

The other fields in the Size Estimating Template will be updated automatically.

3.4.3.9 PSP1.1 Project Plan Summary. The following steps describe the usage of the PSP1.1 Project Plan Summary.

1. When the user loads this applet for the first time, it will show "User Name", "Date", and "PSP Progress".
2. Enter "Project Name".
3. Enter "Supervisor".
4. Enter "Language", for example, C++ or Java.
5. Update "Date" with the format of "YYYY-MM-DD". The user must enter the exact date in order to calculate "To Date" field in the future.
6. Enter plan "Base", "Deleted", and "Modified" LOC.
7. Enter actual "Base", "Deleted", "Modified", and "Total" LOC with the actual LOC for the program the user wants to record.
8. Enter plan "Planning", "Design", "Code", "Compile", "Test", and "Postmortem" time.

9. Click "Calculate & Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then the PSP Scriber has finished the database update. Some fields of this form will be calculated automatically at this point.
10. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields (e.g. defect injected or defect removed) in the PSP1.1 Project Plan Summary will be updated automatically after the user has inserted records on the "Time Recording Log" and the "Defect Recording Log".

3.4.3.10 Task Planning Template. In the Task Planning Template, the user must enter data on the first column "#", for every row if these rows have data on it. The following steps describe the usage of the Task Planning Template.

1. Enter task "#" with integer.
2. Enter task name.
3. Enter plan hours.

4. Enter date Monday.

5. Enter actual date.

6. Repeat steps 1 through 5 if the user needs to add more rows in Task Planning Template.

7. Click "Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then the PSP Scriber has finished the database update.

8. Click "Print Form" button if the user need to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields in the Task Planning Template will be updated automatically.

3.4.3.11 Schedule Planning Template. In the Schedule Planning Template, user must enter data on the first column "Week No.", for every row if that has data on it. The following steps describe the usage of the Schedule Planning Template.

1. Enter "Week No." with integer.

2. Enter "Date Monday".

3. Enter plan "direct hours".

4. Enter actual "direct hours".

5. Repeat steps 1 through 4 if the user needs to add more rows in the Schedule Planning Template.

6. Click "Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then the PSP Scriber has finished the database update.

7. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields in the Schedule Planning Template will be updated automatically.

3.4.3.12 PSP2 Project Plan Summary. The following steps describe the usage of the PSP2 Project Plan Summary.

1. When the user loads this applet for the first time, it will show "User Name", "Date", and "PSP Progress".

2. Enter "Project Name".

3. Enter "Supervisor".

4. Enter "Language", for example, C++ or Java.

5. Update "Date" in the format of "YYYY-MM-DD". The user must enter the exact date in order to calculate "To Date" field in the future.
6. Enter plan "Base", "Deleted", and "Modified" LOC.
7. Enter actual "Base", "Deleted", "Modified", and "Total" LOC with the actual LOC for the program the user wants to record.
8. Enter plan "Planning", "Design", "Design Review", "Code", "Code Review", "Compile", "Test", and "Postmortem" time.
9. Click "Calculate & Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then the PSP Scriber has finished the database update. Some fields of this form will be calculated automatically at this point.
10. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields (e.g. defect injected or defect removed) in the PSP2 Project Plan Summary will be updated

automatically after the user has inserted records in the "Time Recording Log" and the "Defect Recording Log".

3.4.3.13 PSP2.1 Project Plan Summary. The following steps describe the usages of the PSP2.1 Project Plan Summary.

1. When the user loads this applet for the first time, it will show "User Name", "Date", and "PSP Progress".
2. Enter "Project Name".
3. Enter "Supervisor".
4. Enter "Language", for example, C++ or Java.
5. Update "Date" in the format of "YYYY-MM-DD". The user must enter the exact date in order to calculate "To Date" field in the future.
6. Enter plan "Base", "Deleted", and "Modified" LOC.
7. Enter actual "Base", "Deleted", "Modified", and "Total" LOC with the actual LOC for the program the user wants to record.
8. Enter plan "Planning", "Design", "Design Review", "Code", "Code Review", "Compile", "Test", and "Postmortem" time.

9. Click "Calculate & Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then the PSP Scriber has finished the database update. Some fields of this form will be calculated automatically at this point.
10. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields (e.g. defect injected or defect removed) in the PSP2.1 Project Plan Summary will be updated automatically after the user has inserted records in the "Time Recording Log" and the "Defect Recording Log".

3.4.3.14 PSP3 Project Plan Summary. The following steps describe the usage of the PSP3 Project Plan Summary.

1. When the user loads this applet for the first time, it will show "User Name", "Date", and "PSP Progress".
2. Enter "Project Name".
3. Enter "Supervisor".

4. Enter "Language", for example, C++ or Java.
5. Update "Date" in the format of "YYYY-MM-DD". The user must enter the exact date in order to calculate "To Date" field in the future.
6. Enter plan "Base", "Deleted", and "Modified" LOC.
7. Enter actual "Base", "Deleted", "Modified", and "Total" LOC with the actual LOC for the program the user wants to record.
8. Enter plan "Planning", "High-Level Design", "High-Level Design Review", "Detailed Design", "Detailed Design Review", "Code", "Code Review", "Compile", "Test", and "Postmortem" time.
9. Click "Calculate & Update" button for updating PSP database. If PSP Scriber does not show any error message and displays a confirm dialog box, then the PSP Scriber has finished the database update. Some fields of this form will be calculated automatically at this point.
10. Click "Print Form" button if the user needs to print this form with HTML format. The user can print the form later after he/she finishes this whole assignment kit.

The other fields (e.g. defect injected or defect removed) in the PSP3 Project Plan Summary will be updated automatically after the user has inserted records in the "Time Recording Log" and the "Defect Recording Log".

CHAPTER FOUR

CONCLUSION AND FUTURE DIRECTIONS

4.1 Conclusion

The purpose of PSP Scriber is to help software engineers use Personal Software Process strategy. Without the PSP Scriber, software engineers need to enter data for each form manually. Also, they may encounter a hard time for keeping data correctly for future use. With PSP Scriber, software engineers will only need to learn how to input data and focus more on acquiring the skills for PSP. All mathematical calculations are handled automatically by the PSP Scriber. Moreover, with the centralized database system and Web-based tool, software engineers can check and update their personal data online.

PSP Scriber supports cross-references for the entire assignment kits. When a user inputs data in one particular form, the same data can be referenced by other forms.

PSP Scriber was tested by students in the course CSCI655 Software Engineering in Fall 2001. From the test, many errors and faults were fixed.

The PSP Scriber is composed of five major components: HTML forms, Java Server Pages forms, the Java Bean-based JDBC classes, the Java based PSP applets, and PSP database. Before I start this Master project, these components are all brand new to me. I knew some but not deeply understand each of these technologies.

For HTML forms, I have learned some basic and advance HTML tags for outputting a nice format of HTML pages. I also learned how to use Cascading Style Sheets for each pages to make every page have the same style.

For Java Server Pages forms, I have learned how to manage Java Server Page Web server, like Tomcat, and how to combine simple HTML tags with Java Server Pages for outputting dynamic PSP form pages. Also, accessing Java Bean_based JDBC classes from Java Server pages are major highlights of what I have learned in this project.

Using JDBC to connect Oracle database is one useful function I have learned in this project. I have learned how to control JDBC classes to access PSP database for better efficiency.

For Java_based PSP applets, I have learned how to handle GUI based programming, not only console line based programming.

For the PSP database, I have learned the basic management skill for Oracle database. Also, I learned the Structured Query Language (SQL) for building tables for use by PSP database.

Overall, this project is the largest program I have ever written about 10,000 Java source code. I have to learned how to manage the design, planning, coding, testing and postmortem in order to meet the requirements of the PSP Scriber.

4.2 Future Directions

PSP Scriber is a complete Web based tool. There are some disadvantages.

- Using Java applets, for security reason, Oracle database does not allow Java applets that are not located in the same server to access the database.
- Using Java applets, result in slower performance in comparison to other solutions.
- The PSP Scriber must access the database for retrieving lots of data. Since users have analog

modems for connection to the Internet, it takes a lot of time to download data and files.

- There is no convenient help system in the PSP Scriber.

These are some of the major disadvantages obtained from the test by student in CSCI 655. In solving these problems, the next section will describe the future directions of PSP Scriber.

4.3 Using Java Server Pages

PSP Scriber uses Java applets to provide the data input, display, calculation, and update. An alternative approach is to use Java Server Pages as the output layout. Inside the Java Server Pages, same as PSP applets, Java-bean based JDBC classes can perform the functions of calculation and update of the database.

This alternative approach has two advantages. The first one is that it will save time by not loading all the Java applets. Second, in this architecture, Web server may be located in a different server as the Oracle database.

4.4 Window Based Application

The PSP Scriber is a Web-based tool. An alternative approach is to design a whole new tool that is Window-

based application. Instead of using database management system to record the data, it saves the data in a single file with its own format. Users can install the application into their own local computer. This will allow users to input, update, and print data from their own site.

This alternative approach has one advantage. The user will save a lot time by not loading the entire database via the Internet.

4.5 Help System

The PSP Scriber provides all the instructions for every form, template, and standards. However, the user still needs to back and forth some pages for looking those instructions. The PSP Scriber lacks convenient help system. An alternative approach is the PSP Scriber will provide help system with instructions inside the PSP applet or the page the user is working on. This solution will help the user who is not familiar with PSP to save time working on the assignment kits.

APPENDIX A:

DETAIL TESTING RESULTS

A.1 Kit 1 Testing Result

A.1.1 Expecting Results

Table C14 PSP0 Project Plan Summary

Student	Pauline Braginton	Date	2001-10-01
Program	1A	Program #	0
Instructor	Dr. A. I. Concepcion	Language	C++

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning		15	15	8.93
Design		20	20	11.90
Code		55	55	32.74
Compile		40	40	23.81
Test		20	20	11.90
Postmortem		18	18	10.71
Total	190	168	168	100.00

Defects Injected	**Actual**	**To Date**	**To Date %**
Planning	0	0	0.00
Design	0	0	0.00
Code	12	12	100.00
Compile	0	0	0.00
Test	0	0	0.00
Total Development	12	12	100.00

Defects Removed	**Actual**	**To Date**	**To Date %**
Planning	0	0	0.00
Design	0	0	0.00
Code	0	0	0.00
Compile	8	8	66.67
Test	4	4	33.33
Total Development	12	12	100.00
After Development	0	0	

Table C16 Time Recording Log

Student	Pauline Braginton	Date	2001-10-01
Instructor	Dr. A. I. Concepcion	Program #	1A

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
09/26	09:00	09:15	0	15	Planning	
09/26	09:15	09:35	0	20	Design	
09/28	09:00	09:55	0	55	Code	
09/29	04:00	04:40	0	40	Compile	
09/29	04:40	05:00	0	20	Test	
09/30	09:00	09:18	0	18	PM	

Table C18 Defect Recording Log

Student	Pauline Braginton	Date	2001-10-01
Instructor	Dr. A. I. Concepcion	Program #	1A

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
09/30	1	Syntax	Code	Compile	1	

Description: Punctuation, missing a colon

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/01	2	Assign	Code	Compile	1	

Description: Declaration, data type mismatch.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/01	3	Assign	Code	Compile	6	

Description: Duplicate name, function with variable.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/01	4	Syntax	Code	Compile	1	

Description: Punctuation, missing parenthesis.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/04	5	Syntax	Code	Compile	2	

Description: Spelling, mismatch variable name.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/04	6	Syntax	Code	Compile	2	

Description: Misspelled variable

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/04	7	Function	Code	Compile	10	

Description: Mistake in the formula

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/04	8	Syntax	Code	Compile	1	

Description: Missing semicolon

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/06	9	Syntax	Code	Test	2	

Description: Undeclared variable.

Table C18 Defect Recording Log

Student: Pauline Braginton
Date: 2001-10-01
Instructor: Dr. A. I. Concepcion
Program #: 1A

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/06	10	Syntax	Code	Test	2	

Description: Variable was not initialized.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/06	11	Syntax	Code	Test	1	

Description: Missing semicolon.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/06	12	Syntax	Code	Test	1	

Description: Syntax spelling

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

A.1 2 Actual Results

PSP0 Project Plan Summary

Employee:	Pauline Braginton	**Date:**	2001-10-01
Project Name:	1A	**PSP Progress:**	0.0
Supervisor:	Instructor: Dr . A . I . Concepcion	**Language:**	C++

Time in Phase (min.)	Plan	Actual	To Date	To Date %
Planning		15	15	8.93
Design		20	20	11.90
Code		55	55	32.74
Compile		40	40	23.81
Test		20	20	11.90
Postmortem		18	18	10.71
Total	190	168	168	100.00

Defects Injected	Actual	To Date	To Date %
Planning	0	0	0.00
Design	0	0	0.00
Code	12	12	100.00
Compile	0	0	0.00
Test	0	0	0.00
Total Development	12	12	100.00

Defects Removed	Actual	To Date	To Date %
Planning	0	0	0.00
Design	0	0	0.00

Code	0	0	0.00
Compile	8	8	66.67
Test	4	4	33.33
Total Development	12	12	100.00
After Development	0	0	

Time Recording Log

Employee:	Pauline Braginton	**Date:**	2001-10-01
Project Name:	1A	**PSP Progress:**	0.0
Supervisor:	Instructor: Dr . A . I . Concepcion	**Language:**	C++

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
2001-09-26	09:00	09:15	0	15	Planning	
2001-09-26	09:15	09:35	0	20	Detail design	
2001-09-28	09:00	09:55	0	55	Code	
2001-09-29	04:00	04:40	0	40	Compile	
2001-09-29	04:40	05:00	0	20	Test	
2001-09-30	09:00	09:18	0	18	Postmortem	

Defect Recording Log

Employee:	Pauline Braginton	**Date:**	2001-10-01
Project Name:	1A	**PSP Progress:**	0.0
Supervisor:	Instructor: Dr . A . I . Concepcion	**Language:**	C++

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-09-30	1	Syntax	Code	Compile	1	0
Description:	punctuation, missing a colon					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-01	2	Assignment	Code	Compile	1	0
Description:	declaration, data type mismatch					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-01	3	Assignment	Code	Compile	6	0
Description:	duplicate name, function with variable					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-01	4	Syntax	Code	Compile	1	0
Description:	punctuation, missing parenthesis					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-04	5	Syntax	Code	Compile	2	0
Description:	spelling, mismatch variable names					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-04	6	Syntax	Code	Compile	2	0

Description:	misspelled variable

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-04	7	Function	Code	Compile	10	0

Description:	mistake in the formula

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-04	8	Syntax	Code	Compile	1	0

Description:	missing semicolon

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-06	9	Syntax	Code	Test	2	0

Description:	undeclared variable

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-06	10	Syntax	Code	Test	2	0

Description:	variable was not initialized

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-06	11	Syntax	Code	Test	1	0

Description:	missing semicolon

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-06	12	Syntax	Code	Test	1	0

Description:	syntax spelling

A.2 Kit 4 Testing Result

A.2.1 Expected Results

Table C34 PSP1 Project Plan Summary

Student	Pauline Braginton	Date	2001-10-20
Program	4A	Program #	4A
Instructor	Dr. A. I. Concepcion	Language	C++

Summary	*Plan*	*Actual*	*To Date*
LOC/Hour	*24.558*	*34.857*	*29.307*

Program Size (LOC):	***Plan***	**Actual**	**To Date**
Base(B)	0	0	
	(Measured)	(Measured)	
Deleted (D)	0	0	
	(Estimated)	(Counted)	
Modified (M)	0	0	
	(Estimated)	(Counted)	
Added (A)	88	122	
	(N-M)	(T-B+D-R)	
Reused (R)	0	0	0
	(Estimated)	(Counted)	
Total New & Changed (N)	88	122	423
	(Estimated)	(A+M)	
Total LOC (T)	88	122	485
	(N+B-M-D+R)	(Measured)	
Total New Reused	0	0	0

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning	21	20	73	8.43
Design	20	20	73	8.43
Code	59	60	260	30.02
Compile	42	40	160	18.48
Test	22	20	190	21.94
Postmortem	51	50	110	12.70
Total	215	210	866	100.0

Defects Injected	**Actual**	**To Date**	**To Date %**
Planning	0	0	0.0
Design	0	4	12.9
Code	5	27	87.10
Compile	0	0	0.0
Test	0	0	0.0
Total Development	5	31	100.0

Defects Removed	**Actual**	**To Date**	**To Date %**

Planning	0	0	0.0
Design	0	1	3.23
Code	0	0	0.0
Compile	4	19	61.29
Test	1	11	35.48
Total Development	5	31	100.00
After Development	0	0	

Table C37 Test Report Template

Student	Pauline Braginton	Date	2001-10-20
Instructor	Dr. A. I. Concepcion	Program #	4A

Test Name/Number	Linear Regression test #1
Test Objective	Test prog.4A. that calculates Linear Regression parameters Beta0 and Beta1.
Test Description	In a file called "file1.cpp" I enter a given set of historical data for variables x and y. Where xi are variables for Estimated Object LOC and yi are variable for Actual New and Changed LOC. When testing the program, we get the values for B0 and B1. Then, I compared the output with the expected results and fill in the test result form.
Test Conditions	A sufficient data, file1.cpp, must exist in order to produce a statistically significant results. file1.cpp is use to test the data. I must contain n sets of historical data for x and y (Estimated Object LOC(x), and Actual New and change(y)).
Expected Results	Beta0 = -22.55 Beta1 = 1.72
Actual Results	Beta0 = -22.5525 Beta1 = 1.72793

Test Name/Number	Linear Regression test#2:
Test Objective	Test prog.4A that calculates Linear Regression parameters Beta0 and beta1.
Test Description	In a file called "file2.cpp" I enter a given set of historical data for variables x and y. Where xi are variables for Estimated New and Changed LOC and yi are variables for Actual New and Changed LOC. When testing the program, we get the values for B0 and B1. I compare the output with the expected results and fill in the test result form.
Test Conditions	A sufficient data, file2.cpp, must exist in order to produce a statistically significant result. File2.cpp is used to test the data. It must n sets of historical data for x and y (Estimated New and Changed LOC(x), and Actual New and Changed LOC (y)).
Expected Results	Beta0 = -23.92 Beta1 = 1.43
Actual Results	Beta0 = -23.9239 Beta1 = 1.43097

Table C39 Size Estimating Template

Student	Pauline Bragiton	Date	2001-10-20
Instructor	Dr. A. I. Concepcion	Program #	4A

BASE PROGRAM	ESTIMATE	ACTUAL
BASE SIZE (B) => => => => => => => => => =>	0	0
LOC DELETED (D) => => => => => => => => =>	0	0
LOC MODIFIED (M) => => => => => => => => =>	0	0

PROJECTED LOC

BASE ADDITIONS:	TYPE	METHODS	REL. SIZE	LOC	LOC
TOTAL BASE ADDITIONS (BA) => => => => => =>				0	0

NEW OBJECTS:	TYPE[1]	METHODS	REL. SIZE	LOC (New Reused *)	
Data	Data	0	Small	25	20
Linear Regression	Cal.	5	Medium	80	69
TOTAL NEW OBJECTS (NO) => => => => => => =>				105	89

REUSED PROGRAMS

REUSED TOTAL (R) => => => => => => => => =>	0	0

		SIZE	TIME
Projected LOC:	E = BA+NO+M	105	
Regression Parameter:	β_0(size and time)	19.538	150.877
Regression Parameter:	β_1(size and time)	0.653	0.497
Estimated New and Changed LOC:	N = $\beta_0 + \beta_1$*E	88.116	
Estimated Total LOC:	T = N + B - D - M + R	88.000	
Estimated Total New Reused (sum of * LOC):		0	
Estimated Total Development Time:	Time = $\beta_0 + \beta_1$*E		203.077
Prediction Range:	Range	49.288	57.346
Upper Prediction Interval:	UPI = N + Range	137.404	260.423
Lower Prediction Interval:	LPI = N - Range	38.827	145.731
Prediction Interval Percent:		70%	70%

[1] L-Logic, I-I/O, C-Calculation, T-Text, D-Data, S-Set-up

Table C27 Process Improvement Proposal (PIP)

Student	Pauline Bragiton	Date	2001-10-20
Instructor	Dr. A. I. Concepcion	Program #	4A
Process	PSP 1	Elements	

PIP Number

	Problem Description:
1	When using two classes, there need to be a plan how to connect them together.
2	When more then one variable is needed to be access at the same time, one class cannot be used.

PROPOSAL

PIP #	**Proposal Description**
1	connecting the classes by link it together with the right variables and pass by reference.
2	By creating two classes, for storing data and one for calculations, I can take care of the calculation problem.

Notes and Comments:

It helps to write a detailed plan in the planning phase, in order to connect
between the two classes.

Table C16 Time Recording Log

Student	Pauline Bragiton	Date	2001-10-20
Instructor	Dr. A. I. Concepcion	Program #	4A

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
10/20	09:00	09:20		20	Plan	
10/20	09:30	09:50		20	Design	
10/19	18:00	19:00		60	Coding	
10/19	20:00	20:40		40	compile	
10/19	21:00	21:20		20	Test	
10/30	10:00	10:50		50	PM	

Table C18 Defect Recording Log

Student	Heng-Jui Tsao	Date	2001-10-20
Instructor	Dr. A. I. Concepcion	Program #	4A

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
10/20	1	Syntax	Code	Compile	20	

Description: problem with pointer syntax.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
	2	Syntax	Code	Compile	10	

Description: spelling error of class name and function name.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
	3	Syntax	Code	Compile	7	

Description: duplicate called function name

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
	4	Syntax	Code	Compile	3	

Description: misspelled function name in the main.

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
	5	Function	Code	Compile	15	

Description: mistake with the formula

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

A.2.2 Actual Result

PSP1 Project Plan Summary

Employee:	Pauline Braginton	**Date:**	2001-10-20
Project Name:	4A	**PSP Progress:**	1.0
Supervisor:	Dr. Concepcion	**Language:**	C++

Summary	***Plan***	***Actual***	***To Date***
LOC/Hour	24.558	34.857	29.307

Program size (LOC)	***Plan***	***Actual***	***To Date***
Base (B)	0	0	
		(Measured)	
Deleted (D)	0	0	
		(Counted)	
Modified (M)	0	0	
		(Counted)	
Added (A)	88	122	
		(T - B + D - R)	
Reused	0	0	0
		(Counted)	
Total New & Changed (N)	88	122	423
		(A + M)	
Total LOC (T)	88	122	485
		(Measured)	
Total New Reused	0	0	0

Time in Phase (min.)	**Plan**	**Actual**	**To Date**	**To Date %**
Planning	21	20	73	8.43

Design	20	20	73	8.43
Code	59	60	260	30.02
Compile	42	40	160	18.48
Test	22	20	190	21.94
Postmortem	51	50	110	12.70
Total	215	210	866	100.00

Defects Injected	**Actual**	**To Date**	**To Date %**
Planning	0	0	0.00
Design	0	4	12.90
Code	5	27	87.10
Compile	0	0	0.00
Test	0	0	0.00
Total Development	5	31	100.00

Defects Removed	**Actual**	**To Date**	**To Date %**
Planning	0	0	0.00
Design	0	1	3.23
Code	0	0	0.00
Compile	4	19	61.29
Test	1	11	35.48
Total Development	5	31	100.00
After Development	0	0	

Test Report Template

Employee:	Pauline Braginton	**Date:**	2001-10-20
Project Name:	4A	**PSP Progress:**	1.0
Supervisor:	Dr. Concepcion	**Language:**	C++

Test Name/Number	Linear Regression test #1:
Test Objective	Test prog.4A. that calculates Linear Regression parameters Beta0 and Beta1.
Test Description	In a file called "file1.cpp" I enter a given set of historical data for variables x and y. Where xi are variables for Estimated Object LOC and yi are variable for Actual New and Changed LOC. When testing the program, we get the values for B0 and B1. Then, I compared the output with the expected results and fill in the test result form.
Test Conditions	A sufficient data, file1.cpp, must exist in order to produce a statistically significant results. file1.cpp is use to test the data. I must contain n sets of historical data for x and y (Estimated Object LOC(x), and Actual New and change(y)).
Expected Results	B0 = -22.55 B1 = 1.72
Actual Results	B0 = -22.5525 B1 = 1.72793

Test Name/Number	Linear Regression test#2:
Test Objective	Test prog.4A that calculates Linear Regression parameters Beta0 and beta1.
Test Description	In a file called "file2.cpp" I enter a given set of historical data for variables x and y. Where xi are variables for Estimated New and Changed LOC and yi are variables for Actual New and Changed LOC. When testing the program, we get the values for B0 and B1. I compare the output with the expected results and fill in the test result form.
Test Conditions	A sufficient data, file2.cpp, must exist in order to produce a statistically significant result. file2.cpp is used to test the data. It must n sets of historical data for x and y (Estimated New and Changed LOC(x), and Actual New and Changed LOC (y)).
Expected Results	B0 = -23.92 B1 = 1.43
Actual Results	B0 = -23.9239 B1 = 1.43097

Size Estimating Template

Employee:	Pauline Braginton	**Date:**	2001-10-20
Project Name:	4A	**PSP Progress:**	1.0
Supervisor:	Dr. Concepcion	**Language:**	C++

BASE PROGRAM		ESTIMATE	ACTUAL
BASE SIZE (B)	=> => => => => => => =>	0	0
LOC DELETED (D)	=> => => => => => => =>	0	0
LOC MODIFIED (M)	=> => => => => => => =>	0	0

PROJECTED LOC

BASE ADDITIONS:	TYPE	METHODS	REL. SIZE	LOC	LOC
TOTAL BASE ADDITIONS		=> => => => => => => =>		0	0
NEW OBJECTS:	TYPE1	METHODS	REL. SIZE	LOC (New	Reused *)
Data	Data	0	Small	25	20
LinearRegression	Calculation	5	Medium	80	69
TOTAL NEW OBJECTS (NO)		=> => => => => => => =>		105	89

REUSED PROGRAMS

REUSED TOTAL (R)	=> => => => => => =>	0	0
		SIZE	TIME
Projected LOC:	E = BA + NO + M	105	
Regression Parameter:	β_0(SIZE AND TIME)	19.538	150.877
Regression Parameter:	β_1(SIZE AND TIME)	0.653	0.497
Estimated New and Changed LOC:	$N = \beta_0 + \beta_1 * E$	88.116	
Estimated Total LOC:	T = N + B - D - M + R	88.000	

Estimated Total New Reused (sum of * LOC):		0	
Estimated Total Development Time:	Time = $\beta_0 + \beta_1 * E$		203.077
Prediction Range:	Range	49.288	57.346
Upper Prediction Interval:	UPI = N + Range	137.404	260.423
Lower Prediction Interval:	LPI = N - Range	38.827	145.731
Prediction Interval Percent:		70%	70%

[1]L-Logic, I-I/O, C-Calculation, T-Text, D-Data, S-Set-up

Process Improvement Proposal

Employee:	Pauline Braginton	**Date:**	2001-10-20
Project Name:	4A	**PSP Progress:**	1.0
Supervisor:	Dr. Concepcion	**Language:**	C++

PIP Number	Problem Description:
1	When using two classes, there need to be a plan how to connect them together.
2	When more then one variable is needed to be access at the same time, one class cannot be used.
Proposal PIP #	**Proposal Description:**
1	connecting the classes by link it together with the right variables and pass by reference.
2	By creating two classes, for storing data and one for calculations, I can take care of the calculation problem.

Notes and Comments:

It helps to write a detailed plan in the planning phase, in order to connect between the two classes.

Time Recording Log

Employee:	Pauline Braginton	**Date:**	2001-10-20
Project Name:	4A	**PSP Progress:**	1.0
Supervisor:	Dr. Concepcion	**Language:**	C++

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
2001-10-20	09:00	09:20	0	20	Planning	
2001-10-20	09:30	09:50	0	20	Detail design	
2001-10-19	18:00	19:00	0	60	Code	
2001-10-19	20:00	20:40	0	40	Compile	
2001-10-19	21:00	21:20	0	20	Test	
2001-10-20	10:00	10:50	0	50	Postmortem	

Defect Recording Log

Employee:	Pauline Braginton	**Date:**	2001-10-20
Project Name:	4A	**PSP Progress:**	1.0
Supervisor:	Dr. Concepcion	**Language:**	C++

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-20	1	Syntax	Code	Compile	20	0
Description:	problem with pointer syntax.					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-20	2	Syntax	Code	Compile	10	0
Description:	spelling error of class name and function name.					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-20	3	Syntax	Code	Compile	7	0
Description:	duplicate called function name					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-20	4	Syntax	Code	Compile	3	0
Description:	misspell function name in the main.					

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-10-20	5	Function	Code	Test	15	0
Description:	mistake with the formula					

A.3 Kit 10

A.3.1 Expected results

Table C63 PSP2.1 Project Plan Summary

Student	Pauline Braginton	Date	2001-11-22
Program	9A, Ex10	Program #	9A
Instructor	Dr. Concepcion	Language	C++

Summary	Plan	Actual	To Date
LOC/Hour	24.818	13.556	30.727
Planned Time	220		2060
Actual Time		270	2021
CPI(Cost-Performance Index)			1.019 (Planned/Actual)
% Reused	20.906	22.059	26.203
% New Reused	0.000	0.000	0.000
Test Defects/KLOC	10.989	16.393	15.459
Total Defects/KLOC	32.967	65.574	46.377
Yield %	0.000	50.000	8.333
% Appraisal COQ	22.727	22.222	7.669
% Failure COQ	22.727	24.074	30.925
COQ A/F Ratio	1.000	0.923	0.248

Program Size (LOC):	Plan	Actual	To Date
Base(B)	363 (Measured)	363 (Measured)	
Deleted (D)	0 (Estimated)	0 (Counted)	
Modified (M)	0 (Estimated)	0 (Counted)	
Added (A)	91 (N-M)	61 (T-B+D-R)	
Reused (R)	120 (Estimated)	120 (Counted)	659
Total New & Changed (N)	91 (Estimated)	61 (A+M)	1035
Total LOC (T)	574 (N+B-M-D+R)	544 (Measured)	2515
Total New Reused	0	0	0
Upper Prediction Interval (70%)	140.805		
Lower Prediction Interval (70%)	40.656		

Time in Phase (min.)	Plan	Actual	To Date	To Date %
Planning	15	20	208	10.29
Design	15	20	203	10.04
Design review	25	30	75	3.71
Code	65	75	565	27.96
Code review	25	30	80	3.96
Compile	25	35	285	14.10
Test	25	30	340	16.82
Postmortem	25	30	265	13.11
Total	220	270	2021	100.00
Total Time UPI (70%)	256.296			
Total Time LPI (70%)	156.909			

(continued)

Table C63 PSP2.1 Project Plan Summary (continued)

Student	Pauline Braginton	Date	2001-11-22
Program	9A, Ex10	Program #	9A
Instructor	Dr. Concepcion	Language	C++

Defects Injected	Plan	Actual	To Date	To Date %
Planning	0	0	0	0.00
Design	0	0	4	8.33
Design review	0	0	0	0.00
Code	4	4	44	91.67
Code review	0	0	0	0.00
Compile	0	0	0	0.00
Test	0	0	0	0.00
Total Development	4	4	48	100.00

Defects Removed	Plan	Actual	To Date	To Date %
Planning	0	0	0	0.00
Design	0	0	1	2.08
Design review	0	2	2	4.17
Code	0	0	0	0.00
Code review	0	0	1	2.08
Compile	2	1	28	58.33
Test	1	1	16	33.33
Total Development	3	4	48	100.00
After Development	0	0	0	

Defect Removal Efficiency	Plan	Actual	To Date
Defects/Hour - Design review	0.000	4.000	1.600
Defects/Hour - Code review	0.000	0.000	0.750
Defects/Hour - Compile	4.800	1.714	5.895
Defects/Hour - Test	2.400	2.000	2.824
DRL(DLDR/UT)	0.000	2.000	0.567
DRL(CodeReview/UT)	0.000	0.000	0.266
DRL(Compile/UT)	2.000	0.857	2.088

Table C47 Task Planning Template

Student	Pauline Braginton	Date	2001-11-22
Project	9A	Instructor	Dr. Concepcion

Task		Plan					Actual		
#	Name	Hours	Planned Value	Cumulative Hours	Cumulative Planned Value	Date Monday	Date	Earned Value	Cumulative Earned Value
1	Planning	15.00	6.667	15.00	6.667	11/23	11/23	6.667	6.667
2	Detail Design	15.00	6.667	30.00	13.333	11/23	11/23	6.667	13.333
3	Detail Design Review	25.00	11.111	55.00	24.444	11/23	11/23	11.111	24.444
4	Code	65.00	28.889	120.00	53.333	11/23	11/23	28.889	53.333
5	Code Review	25.00	11.111	145.00	64.444	11/24	11/24	11.111	64.444
6	Compile	30.00	13.333	175.00	77.778	11/24	11/24	13.333	77.778
7	Test	25.00	11.111	200.00	88.889	11/24	11/24	11.111	88.889
8	Postmortem	25.00	11.111	225.00	100.00	11/24	11/24		
Totals		250	100.0						

Table C49 Schedule Planning Template

Student	Pauline Braginton	Date	2001-11-22
Project	9A, Ex10	Instructor	Dr. Concepcion

		Plan			Actual			
Week No.	Date Monday	Direct Hours	Cumulative Hours	Cumulative Planned Value	Direct Hours	Cumulative Hours	Cumulative Earned Value	Adjusted Earned Value
1	11/23	15.00	15.00	6.667	20.00	20.00	8.889	
2	11/23	15.00	30.00	13.333	20.00	40.00	17.778	
3	11/23	25.00	55.00	24.444	30.00	70.00	31.111	
4	11/23	65.00	120.00	53.333	75.00	145.00	64.444	
5	11/24	25.00	145.00	64.444	30.00	175.00	77.778	
6	11/24	30.00	175.00	77.778	35.00	210.00	93.333	
7	11/24	25.00	200.00	88.889	30.00	240.00	106.667	
8	11/24	25.00	225.00	100.000				

Table C37 Test Report Template

Student	Pauline Braginton	Date	2001-11-22
Instructor	Dr. A. I. Concepcion	Program #	9A

Test Name/Number	**X^2 Distribution #1**
Test Objective	**To calculate the values X^2 distribution**
Test Description	**Use "D14.cpp" as an input file. and compare the output for LOC/Method(y), with the expected value in the book.**
Test Conditions	**Iput file from table D14.**
Expected Results	**Q=34.4 (1-P)=7.60*10^-5**
Actual Results	**Q=34.4 (1-P)=7.60*10^-5**

Test Name/Number	**X^2 Distribution Test#2**
Test Objective	**Calculate the x^2 distribution**
Test Description	**File "D14.cpp" used as an input file. In test #2 we check the test result for Object LOC(x).**
Test Conditions	**Use input file from table D14, from the text book.**
Expected Results	**Q=49 (1-p)=0.000124**
Actual Results	**Q=49.6 (1-p)=0.000124248**

Table C39 Size Estimating Template

Student	Pauline Bragiton	Date	2001-10-20
Instructor	Dr. A. I. Concepcion	Program #	9A

BASE PROGRAM				ESTIMATE	ACTUAL
BASE SIZE (B) => => => => => => => => => =>				363	363
LOC DELETED (D) => => => => => => => => =>				0	0
LOC MODIFIED (M) => => => => => => => => =>				0	0
PROJECTED LOC					
BASE ADDITIONS:	TYPE	METHODS	REL. SIZE	LOC	LOC
List	Cal.	1	Medium	12	14
Sort	Cal.	5	Medium	56	47
TOTAL BASE ADDITIONS (BA) => => => => => =>				68	61
NEW OBJECTS:	TYPE[1]	METHODS	REL. SIZE	LOC (New Reused *)	
TOTAL NEW OBJECTS (NO) => => => => => => =>					
REUSED PROGRAMS					
Integration				120	120
REUSED TOTAL (R) => => => => => => => => =>				120	120

		SIZE	TIME
Projected LOC:	E = BA+NO+M	68	
Regression Parameter:	β_0(size and time)	51.859	177.840
Regression Parameter:	β_1(size and time)	0.572	0.423
Estimated New and Changed LOC:	$N = \beta_0 + \beta_1 * E$	90.731	
Estimated Total LOC:	T = N + B - D - M + R	574.000	
Estimated Total New Reused (sum of * LOC):		0	
Estimated Total Development Time:	$Time = \beta_0 + \beta_1 * E$		206.603
Prediction Range:	Range	50.075	49.693
Upper Prediction Interval:	UPI = N + Range	140.805	256.296
Lower Prediction Interval:	LPI = N - Range	40.656	156.909
Prediction Interval Percent:		70%	70%

[1] L-Logic, I-I/O, C-Calculation, T-Text, D-Data, S-Set-up

Table C27 Process Improvement Proposal (PIP)

Student	Pauline Braginton	Date	2001-11-22
Instructor	Dr. A. I. Concepcion	Program #	9A
Process	2.1	Elements	

PIP Number

Problem Description:

1 The List that hold values did not get sorted

PROPOSAL

PIP # **Proposal Description**

1 I had to modify the List file "List3.cpp". Added another sort function to the class List.

Notes and Comments:

Table C16 Time Recording Log

Student	Pauline Braginton	Date	2001-11-22
Instructor	Dr. Concepcion	Program #	9A

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
11/23	09:00	09:20		20	Plan	
11/23	09:20	09:40		20	Detail Design	
11/23	10:00	10:30		30	Detail Design Review	
11/23	11:00	12:15		75	Code	
11/24	09:00	09:30		30	Code Review	
11/24	09:30	10:05		35	Compile	
11/24	10:05	10:35		30	Test	
11/24	10:35	11:05		30	PM	

Table C18 Defect Recording Log

Student	Heng-Jui Tsao	Date	2001-11-22
Instructor	Dr. A. I. Concepcion	Program #	9A

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
11/22	1	Function	Code	Test	20	

Description: Mistake in the formula

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
11/22	2	Syntax	Code	Dd review	5	

Description: missing a colon

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
11/22	3	Syntax	Code	Compile	5	

Description: Variable mismatch

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
11/22	4	Syntax	Code	Dd review	5	

Description: Misspelled function name

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect

Description:

A.3.2 Actual Result

PSP2.1 Project Plan Summary

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

Summary	**Plan**	**Actual**	**To Date**
LOC/Hour	24.818	13.556	30.727
Planned Time	220		2060
Actual Time		270	2021
CPI(Cost-Performance Index)			1.019
			(Planned/Actual)
% Reused	20.906	22.059	26.203
% New Reused	0.000	0.000	0.000
Test Defects/KLOC	10.989	16.393	15.459
Total Defects/KLOC	32.967	65.574	46.377
Yield%	0.000	50.000	8.333
% Appraisal COQ	22.727	22.222	7.669
% Failure COQ	22.727	24.074	30.925
COQ A/F Ratio	1.000	0.923	0.248

Program size (LOC)	***Plan***	***Actual***	***To Date***
Base (B)	363	363	
		(Measured)	
Deleted (D)	0	0	
		(Counted)	
Modified (M)	0	0	
		(Counted)	
Added (A)	91	61	

		(T - B + D - R)	
Reused	120	120	659
		(Counted)	
Total New & Changed (N)	91	61	1035
		(A + M)	
Total LOC (T)	574	544	2515
		(Measured)	
Total New Reused	0	0	0
Upper Prediction Interval (70%)	140.805		
Lower Prediction Interval (70%)	40.656		

Time in Phase (min.)	Plan	Actual	To Date	To Date %
Planning	15	20	208	10.29
Design	15	20	203	10.04
Design Review	25	30	75	3.71
Code	65	75	565	27.96
Code Review	25	30	80	3.96
Compile	25	35	285	14.10
Test	25	30	340	16.82
Postmortem	25	30	265	13.11
Total	220	270	2021	100.00
Total Time UPI (70%)	256.296			
Total Time LPI (70%)	156.909			

Defects Injected	Plan	Actual	To Date	To Date %
Planning	0	0	0	0.00
Design	0	0	4	8.33
Design Review	0	0	0	0.00

Code	4	4	44	91.67
Code Review	0	0	0	0.00
Compile	0	0	0	0.00
Test	0	0	0	0.00
Total Development	4	4	48	100.00

Defects Removed	**Plan**	**Actual**	**To Date**	**To Date %**
Planning	0	0	0	0.00
Design	0	0	1	2.08
Design Review	0	2	2	4.17
Code	0	0	0	0.00
Code Review	0	0	1	2.08
Compile	2	1	28	58.33
Test	1	1	16	33.33
Total Development	3	4	48	100.00
After Development	0	0	0	

Defect Removal Efficiency	**Plan**	**Actual**	**To Date**
Defcts/Hour-Design Review	0.000	4.000	1.600
Defects/Hour-Code Review	0.000	0.000	0.750
Defects/Hour-Compile	4.800	1.714	5.895
Defects/Hour-Test	2.400	2.000	2.824
DRL(DLDR/UT)	0.000	2.000	0.567
DRL(CodeReview/UT)	0.000	0.000	0.266
DRL(Compile/UT)	2.000	0.857	2.088

Task Planning Template

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

Task		Plan					Actual		
#	Name	Hours	Planned Value	Cumulative Hours	Cumulative Planned Value	Date Monday	Date	Earned Value	Cumulative Earned Value
1	Planning	15.00	6.667	15.00	6.667	2001-11-23	2001-11-23	6.667	6.667
2	Detail design	15.00	6.667	30.00	13.333	2001-11-23	2001-11-23	6.667	13.333
3	Detail design review	25.00	11.111	55.00	24.444	2001-11-23	2001-11-23	11.111	24.444
4	Code	65.00	28.889	120.00	53.333	2001-11-23	2001-11-23	28.889	53.333
5	Code review	25.00	11.111	145.00	64.444	2001-11-24	2001-11-24	11.111	64.444
6	Compile	30.00	13.333	175.00	77.778	2001-11-24	2001-11-24	13.333	77.778
7	Test	25.00	11.111	200.00	88.889	2001-11-24	2001-11-24	11.111	88.889
8	Postmortem	25.00	11.111	225.00	100.000	2001-11-24			
Totals		225.00	100.000						

Schedule Planning Template

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

		Plan			Actual			
Week No.	Date Monday	Direct Hours	Cumulative Hours	Cumulative Planned Value	Direct Hours	Cumulative Hours	Cumulative Earned Value	Adjusted Earned Value
1	2001-11-23	15.00	15.00	6.667	20.00	20.00	8.889	
2	2001-11-23	15.00	30.00	13.333	20.00	40.00	17.778	
3	2001-11-23	25.00	55.00	24.444	30.00	70.00	31.111	
4	2001-11-23	65.00	120.00	53.333	75.00	145.00	64.444	
5	2001-11-24	25.00	145.00	64.444	30.00	175.00	77.778	
6	2001-11-24	30.00	175.00	77.778	35.00	210.00	93.333	
7	2001-11-24	25.00	200.00	88.889	30.00	240.00	106.667	
8	2001-11-24	25.00	225.00	100.000				

Test Report Template

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

Test Name/Number	X^2 Distribution #1
Test Objective	To calculate the values X^2 distribution
Test Description	Use "D14.cpp" as an input file. and compare the output for LOC/Method(y), with the expected value in the book.
Test Conditions	Iput file from table D14.
Expected Results	Q=34.4 (1-P)=7.60*10^-5
Actual Results	Q=34.4 (1-P)=7.60*10^-5

Test Name/Number	X^2 Distribution Test#2
Test Objective	Calculate the x^2 distribution
Test Description	File "D14.cpp" used as an input file. In test #2 we check the test result for Object LOC(x).
Test Conditions	Use input file from table D14, from the text book.
Expected Results	Q=49 (1-p)=0.000124
Actual Results	Q=49.6 (1-p)=0.000124248

Size Estimating Template

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

BASE PROGRAM		ESTIMATE	ACTUAL
BASE SIZE (B)	=> => => => => => => =>	363	363
LOC DELETED (D)	=> => => => => => => =>	0	0
LOC MODIFIED (M)	=> => => => => => => =>	0	0

PROJECTED LOC

BASE ADDITIONS:	TYPE	METHODS	REL. SIZE	LOC	LOC
List	Calculation	1	Medium	12	14
sort	Calculation	5	Medium	56	47
TOTAL BASE ADDITIONS		=> => => => => => => =>		68	61

NEW OBJECTS:	TYPE1	METHODS	REL. SIZE	LOC (New Reused *)	
TOTAL NEW OBJECTS (NO)		=> => => => => => => =>		0	0

REUSED PROGRAMS

Integration	120	120
REUSED TOTAL (R) => => => => => =>	120	120

		SIZE	TIME
Projected LOC:	E = BA + NO + M	68	
Regression Parameter:	β_0(SIZE AND TIME)	51.849	177.840
Regression Parameter:	β_1(SIZE AND TIME)	0.572	0.423
Estimated New and Changed LOC:	N = $\beta_0 + \beta_1 * E$	90.731	

Estimated Total LOC:	T = N + B - D - M + R	574.000	
Estimated Total New Reused (sum of * LOC):		0	
Estimated Total Development Time:	Time = $\beta_0 + \beta_1 * E$		206.603
Prediction Range:	Range	50.075	49.693
Upper Prediction Interval:	UPI = N + Range	140.805	256.296
Lower Prediction Interval:	LPI = N - Range	40.656	156.909
Prediction Interval Percent:		70%	70%

[1]L-Logic, I-I/O, C-Calculation, T-Text, D-Data, S-Set-up

Process Improvement Proposal

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

PIP Number	Problem Description:
1	The List that hold values did not get sorted
Proposal PIP #	**Proposal Description:**
1	I had to modify the List file "List3.cpp". Added another sort function to the class List.

Notes and Comments:

Time Recording Log

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

Date	Start	Stop	Interruption Time	Delta Time	Phase	Comments
2001-11-23	09:00	09:20	0	20	Planning	
2001-11-23	09:20	09:40	0	20	Detail design	
2001-11-23	10:00	10:30	0	30	Detail design review	
2001-11-23	11:00	12:15	0	75	Code	
2001-11-24	09:00	09:30	0	30	Code review	
2001-11-24	09:30	10:05	0	35	Compile	
2001-11-24	10:05	10:35	0	30	Test	
2001-11-24	10:35	11:05	0	30	Postmortem	

Defect Recording Log

Employee:	Pauline Braginton	**Date:**	2001-11-22
Project Name:	9A, EX10	**PSP Progress:**	2.1
Supervisor:	Dr. Concepcion	**Language:**	C++

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-11-22	1	Function	Code	Test	20	0

Description: Mistake in the formula

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-11-22	2	Syntax	Code	Detail design review	5	0

Description: missing a colon

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-11-22	3	Syntax	Code	Compile	5	0

Description: Variable mismatch

Date	Number	Type	Inject	Remove	Fix Time	Fix Defect
2001-11-22	4	Syntax	Code	Detail design review	5	0

Description: Misspelled function name

REFERENCES

[1] *Object-Oriented Modeling and Design for Database Applications,* by Michael Blaha and William Premerlani, Prentice-Hall Inc. ISBN 0-13-123829-9.

[2] *UML Distilled 2nd Edition: A Brief Guide to The Standard Object Modeling Language,* by Martin Fowler and Kendall Scott, Addison Wesley Longman, Inc. ISBN 0-201-65783-X.

[3] *Graphic Java 2: Mastering the JFC Volumn II - Swing,* by David M. Geary, Sun Microsystem, Inc. ISBN 0-13-079667-0.

[4] *Core Java 2: Volumn I - Fundamentals,* by Cay S. Horstmann and Gary Cornell, Sun Microsystem, Inc. ISBN 0-13-081933-6.

[5] *A Discipline for Software Engineering,* by Watts S. Humphrey, Addison-Wesley Publishing Company, Inc. ISBN 0-201-54610-8.

[6] IEEE Std. 830-1993, IEEE Recommended Practice for Software Requirements Specifications.

[7] *Introduction to Java Programming with Jbuilder 3,* by Y. Daniel Liang, Prentice-Hall Inc. ISBN 0-13-086911-2.

[8] *Rapid Java Application Development Using Jbuilder 3*, by Y. Daniel Liang, Prentice-Hall Inc. ISBN 0-13-026161-0.

[9] *Oracle 8 Programming: A Primer*, by Rajshekhar Sunderraman, Addison Wesley Longman, Inc. ISBN 0-201-61258-5.

[10] *JDBC API Tutorial and Reference, 2nd Edition*, by Seth White, Maydene Fisher, Rick Cattell, Graham Hamilton, and Mark Hapner, Sun Microsystem, Inc. ISBN 0-201-43328-1.