A PCSE (Practice Centered Software Engineering) tool for Eclipse environment

by

Prathap Subramanian

A thesis submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Auburn, Alabama
May 9, 2011

Keywords: software process, PCSE, eclipse plug-ins,
timelog, changelog, size matrix

Approved by

David A. Umphress, Chair, Associate Professor of Computer Science & Software Engineering
James H. Cross II, Professor of Computer Science & Software Engineering
Jeffrey S. Smith, Professor of Industrial & Systems Engineering

Abstract

Practice Centered Software Engineering (PCSE) is the most recent incarnation of Auburn University's personal self-improvement process for helping software engineers control, manage, and improve the way they work. It helps them make accurate plans, consistently meet commitments, improve QPPC (Quality, Predictability, Productivity, Customer satisfaction), and deliver high-quality products. PCSE is a tailored collection of different elements from various software processes such as Personal Software Process (PSP), Team Software Process (TSP), Extreme Programming (XP), Feature Driven Development (FDD), SCRUM, Rational Unified Process (RUP) etc. PCSE was developed by Dr. David A. Umphress, Department of Computer Science & Software Engineering, Auburn University, in an effort to bring engineering discipline to one-person software development teams.

The objective of this thesis is to develop a set of PCSE plug-ins that are integrated into the Eclipse environment. The plug-ins help an Eclipse user conveniently practice PCSE by reducing data gathering and analysis overhead. They provide automated process steps to collect metrics, user-friendly Eclipse views to input data, process dashboard to display and perform tasks using interactive tools that are placed together in an Eclipse view, support for Save, SaveAs, Open etc. The Eclipse plug-ins are designed to support PCSE time logs, change logs, and size estimation.

Acknowledgments

I take this opportunity to thank all those who helped and guided me throughout this thesis. I consider it a special privilege to convey my prodigious and everlasting thanks to my advisory committee chair, Dr. David A. Umphress, Computer Science & Software Engineering department, Auburn University for all the advice, guidance and support given to me right from the beginning of the thesis. I express my deep sense of gratitude to Dr. James H. Cross II, Computer Science & Software Engineering department, Auburn University, and Dr. Jeffrey S. Smith, Industrial and Systems Engineering department, Auburn University, for their valuable advice, insight, and critical reviews provided throughout my thesis work.

My special thanks to our PCSE research team members, Asmae Mesbahi El Aouame (PhD student), Jackie Hundley (Instructor), Russell Thackston (PhD student), Susan Hammond (PhD student) and Yasmeen Rawajfih (MSwe Student), Computer Science and Software Engineering department, Auburn University, for their support.

With immense pleasure and satisfaction, I express my sincere thanks to all my family members and friends for their kind help and unstinted cooperation and companionship during the thesis work.

Most importantly, I thank and praise the almighty god for giving me strength, courage, guidance, wisdom and knowledge to complete this thesis work successfully.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

PCSE	Practice Centered Software Engineering
PSP	Personal Software Process
TSP	Team Software Process
XP	Extreme Programming
FDD	Feature Driven Development
RUP	Rational Unified Process
QPPC	Quality, Productivity, Predictability, Customer Satisfaction
IDE	Integrated Development Environment
JDT	Java Development Toolkit
LOC	Line of Code
CRC	Class Responsibility Collaborator
LOCr	Raw lines of code
LOCp	Planned lines of code
LOCa	Actual lines of code
Ep	Planned Effort
Ea	Actual Effort
LPI	Lower Prediction Interval
UPI	Upper Prediction Interval
SVN	Apache Subversion

EV	Earned Value
PV	Planned Value
TDD	Test Driven Development
SWT	Standard Widget Toolkit
JNI	Java Native Interface

Introduction

1.1 An Overview:

Software process aims to structure efforts so as to help software engineers understand and improve their performance. That aim can be thwarted if engineers use a process that requires a great deal of manual intervention for capturing and recording process metrics. For example, developers use forms to log time, defects and other metrics; excel worksheets for metrics calculation; etc. This approach leads to the following difficulties:

1.) Decreases productivity of the user.

2.) Decreases the accuracy of the process metrics.

3.) Failure to capture important metrics.

4.) Recording invalid or inaccurate data.

The proposed solution is to develop a set of plug-ins for automating the collection of process metrics for the PCSE process in an Eclipse Integrated Development Environment (IDE). By developing these plug-ins, one can reduce the overhead of manually collecting the process metrics.

1.2 Objective of the Thesis:

The objectives of this thesis are:

1.) To provide an introduction to PCSE and the various artifacts, tools associated with it.

2.) To describe the PCSE Lifecycle.

3.) To explain the different activities of PCSE.

4.) To illustrate PCSE Eclipse plug-ins for: a.) Timelog, b.) Changelog, c.) Size Estimate

1.3 Work breakdown structure:

The following is the work breakdown structure of the subsequent portions of the thesis:

- Chapter 3 (Literature Review) provides details of the PSP automated tools/plug-ins available in the market. It captures in detail the features each tool support and their usage.
- Chapter 4 (The PCSE Life Cycle) explains what PCSE is, the different activities of PCSE, the PCSE Life Cycle, the artifacts and tools associated with it. It also explains in detail how each PCSE artifact is used along with examples, tables and figures.
- Chapter 5 (PCSE Eclipse plug-ins) describes three plug-ins for supporting the most commonly-used elements of PCSE, a.) Timelog, b.) Changelog, c.) Size Estimate. It explains the features and usage of each plug-ins. The scope, implementation, environment and deployment are discussed.
- Chapter 6 (Conclusion & Future Work) documents the scope and assumptions of the plug-ins and their support of important functionalities. It also explores future enhancements that can be done to these plug-ins.

Problem Description

2.1 Software Process:

Software process is the set of tasks needed to produce quality software [1]. It helps developers make accurate plans, consistently meet commitments, improve QPPC (Quality, Predictability, Productivity, Customer satisfaction), and deliver high-quality products. It is a structured framework of forms, guidelines, activities, and procedures for developing software. A growing body of software development organizations implement process methodologies, employing methodologies which are intended to improve software quality, such as Personal Software Process (PSP), Team Software Process (TSP), Extreme Programming (XP), Feature Driven Development (FDD), SCRUM, Rational Unified Process (RUP), etc.

Each of these methodologies describes approaches to a variety of tasks or activities that take place during software development. However, there are no restrictions such that one has to follow a particular methodology while practicing software process. It is generally based on the type of the industry and the nature of the project, apart from other factors such as the budget, technology used, resources, etc.

There are a number of software processes. Personal Software Process (PSP) is the only “recognized” process that addresses one person teams; all others are meant for multiperson efforts. We deal with one person teams. PSP is encumbering due to its requirements for data collection. PCSE is a light-weight alternative to PSP.

2.2 The Personal Software Process (PSP):

The Personal Software Process (PSP) [2] is a structured software development process for a single developer, created by Watts Humphrey in Software Engineering Institute. PSP aims to provide software engineers with disciplined methods for improving personal software development processes that help developers produce zero-defect, quality products on schedule. One of the core aspects of the PSP is using historical data to analyze and improve process performance. [2]

The PSP helps software engineers to:

1.) Improve their estimating and planning skills.

2.) Make commitments they can keep.

3.) Manage the quality of their projects.

4.) Reduce the number of defects in their work.

The following are the advantages of PSP [2]:

1.) It helps the developer understand his performance.

2.) It helps the developer to manage his work.

3.) It helps to plan and manage the quality of products produced.

4.) It helps to make detailed plans and precisely measure and report the status.

5.) It helps to judge the accuracy of your estimates and plans.

6.) It helps to communicate precisely with users, other developers, managers, and customers about the work.

7.) It helps to identify the process steps that cause the most trouble.

8.) It helps to improve the personal performance.

9.) It will simplify training and facilitate personal mobility.

10) Well-defined process definitions can be reused or modified to make new and improved processes.

2.3 The Problem of Practicing Software Process:

Although many software methodologies are available to practice software process, each developer has his own way of working, resulting in inefficient software development. The following are some of the reasons why a process is not effectively practiced.

1.) Developers are not aware of the process itself.

2.) Developers have a tendency to build a working product in an ambiguous way.

3.) Developers treat process as an additional overhead.

4.) Developers have an incomplete understanding of process methodologies.

5.) Developers choose a process which is too heavy for the nature of the project.

6.) Manual data collection by using print out forms or excel worksheets to log effort, size, defect and other information, is an overhead for the developer.

2.4 Need for Light-weight software process:

There are a lot of software development methodologies in use today and the list grows daily. Many developers have their own customized methodology for developing their software, while others use off-the-shelf commercial methodologies. The following factors play an important role in selecting a methodology: budget, team size, project criticality, technology used, documentation, training, tool and techniques etc.

The traditional project methodologies that many developers use are considered to be bureaucratic or predictive in nature, and they've resulted in many unsuccessful projects [3]. They can be so tedious that the whole pace of design, development and deployment actually slows down.

A lightweight software process is a software development methodology that has only a few rules and practices or ones which are easy to follow. It emphasizes the need to deal with changes in requirements and changes in environment or technology by being flexible and adaptive [4]. In lightweight software process, after each build or iteration, the developer learns to correct issues on the project, forming an improvement cycle throughout the project.

The following are the major advantages of lightweight methodologies [3]:

1.) They accommodate change well.

2.) They are people-oriented rather than process-oriented. They tend to work with people rather than against them.

3.) They are complemented by the use of dynamic checklists.

4.) They are much less document oriented.

The number of frequent cycles in the lightweight methodologies also provides more opportunities for developers to review the project definition and redefine it for new business needs. It has room to add new requirements and change the requirements list, adjusting priorities accordingly. Another benefit of the lightweight methodologies is their focus on producing value-added releases and addressing architectural risk early in the project, which would be difficult with a heavyweight methodology.

2.5 Practice Centered Software Engineering (PCSE):

The Practice Centered Software Engineering (PCSE) is a software engineering process developed by Dr. David A. Umphress, Department of Computer Science & Software

Engineering, Auburn University. It is the most recent incarnation of Auburn University's personal self-improvement process that helps software engineers control, manage, and improve the way they work. Using common industry practices, PCSE describes the following activities/phases that are performed within the software development process: Analysis, Architecture, Project plan, Iteration plan, Construction, Review, Refactor, Integration, Post mortem and Code complete. Each of these activities is associated with a particular artifact such as the Operational specification, Scenario-Component map, Iteration map, Conceptual Design, Size matrix, Timelog, Changelog, Iteration map, Burn-down chart, Calendar, Diary etc. The flow of the activities and the use of the artifacts are detailed in the forthcoming chapters.

2.6 Manual Data Collection and the Emergence of Automated Tools:

Process practitioners commonly use page-based forms or spreadsheets to log their effort, product size, defects, and other metrics. Although spreadsheets automate some of the metrics calculation and data analysis, they represent overhead for the developers and become an obstacle in the way of carry out process effectively.

To overcome this obstacle, automated tools such as the PSP Studio [5], the Process Dashboard [6], etc. have been developed to reduce data gathering and analysis overhead. The automated tools are discussed in detail in Chapter 3 (Literature Review) of this thesis.

2.7 About the Eclipse (Software):

Eclipse is a multi-language software development environment consisting of an integrated development environment (IDE) and an extensible plug-in architecture. It is written primarily in Java and can be used to develop applications in Java and, by means of various plug-ins, other languages including C, C++, COBOL, Python, Perl, PHP, Scala, Scheme and Ruby (including Ruby on Rails framework). The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java and Eclipse PDT for PHP [7].

The initial codebase originated from VisualAge [8]. In its default form it is meant for Java developers, and consists of the Java Development Tools (JDT). Users can extend its capabilities by installing plug-ins written for the Eclipse software framework, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Released under the terms of the Eclipse Public License, Eclipse is free and open source software.

Even though Eclipse had a support for multi-language software development, the development of PCSE plug-ins had been restricted to support the Java language.

2.8 PCSE Plug-ins for Eclipse environment:

Although there are a lot of automated tools in the market there is no single tool available for PCSE, as it is unique when compared to other software methodologies. Each of the existing automated tools currently available is different in its nature and not many correspond to Eclipse IDE (Integrated Development Environment). The development of these PCSE Eclipse plug-ins will impact the developer's productivity and result in accurate metrics collection.

Literature Review

The details of automated tools and plug-ins and important literature relevant to individual development are as follows:

3.1 A Personal Software Process Tool for Eclipse Environment [9]:

[9] presents a PSP tool that is integrated into the Eclipse environment. Currently it includes an Eclipse plug-in that supports PSP0 and a Line-Of-Code (LOC) counter for PSP0.1.

3.2 PSP Studio [5]:

Personal Software Process Studio (PSPS) was developed by the 1996-1997 Design Studio team at East Tennessee State University, under the guidance of Dr Joel Henry. It is a separate software program that needs to be installed on a computer before using it [5].

It provides the following features:

a.) ***Facilitates data measurement:*** PSPS measures development time much like a stopwatch and records the data in a log. PSPS also makes recording additional measurements such as defects and size much more convenient.

b.) ***Maintains a historical database:*** PSPS stores all the historical PSP data in a database which insures reliability and security.

c.) ***Provides convenient access to tables:*** Like an engineering notebook, all of the tables in PSPS are indexed by tab controls for convenient access.

d.) ***Performs statistical calculations:*** PSPS automatically maintains historical totals and performs the statistical calculations used in throughout PSP.

e.) ***Provides guidance through the process:*** The automated scripts and the online help system in PSPS provide information and direction for implementing the PSP.

3. 3 Process Dashboard [6]:

The Software Process Dashboard Project is an open-source PSP / TSP support tool, which was originally developed in 1998 by the United States Air Force. It is freely available for download under the conditions of the GNU Public License. It is a separate software program that needs to be installed on any computer before using it [6].

It provides the following features:

a.) ***Data Collection*** - Time, defects, size; plan vs. actual data

b.) ***Planning*** - Integrated scripts, templates, forms, and summaries, PROBE, earned value

c.) ***Tracking*** - Powerful earned value support

d.) ***Data Analysis*** - Charts and reports aid in the analysis of historical data trends.

e.) ***Data Export*** - Export data into Excel, or export data to text format for use with external tools.

3.4 Easy Tracker Lite - Personal Time Manager3.3 [10]:

Easy Tracker Life is a full-featured time tracking suite designed specifically for schedule notification, expense management, work measurement, timesheet entry and project management [10]. It is especially suited for professionals who want to track time to improve personal and team productivity, to manage projects, to manage expenses and budgets, and to monitor projects as work is being done. One can manage all the fundamental information of Clients, Projects, Tasks, Rates and Users by themselves, and assign tasks to them. Easy Tracker Lite provides time tracking method. Just in a mouse click, one can start a time tracking or stop it. All time logs can be viewed in a calendar and grid. And multiple charts are provided to analyze the time distribution. Easy Tracker Lite also provides powerful scheduling functions to assign tasks for themselves. In addition to one-time occurrences, Easy Tracker Lite keeps track of all types of recurring assignments. And it provides multiform reminder methods, such as popping up a notification dialog, sending an email, running an application, playing music, popping up a talking agent or opening a specific web page. If one is too busy, he can hit "snooze" to get that reminder later.

3.5 PSP Log Control [11]:

The basic idea of the process logging is finding out how one spend his/her time and, in particular, how many and what kinds of errors a developer makes, at what time, why, and how long it takes the developer to fix these errors [11].

PSPLog Control [11] helps to collect the appropriate data while programming using any Windows software development tool. With a single mouse click in the task bar, the user can open a menu where he can create entries in his log file with a second mouse click. For convenience, the user do not need to open a dialog window.

3.6 Psptool [12]:

Psptool is a tcl/tk script that runs under X/Unix or an Win32S application. It provides time and defect logging in a convient fashion. It also generates a PSP-2.1-like plan summary sheet giving LOC/h, Defects/KLOC and so forth. The user fill in a few fields as per the usual summary form, and it calculates the rest from the logs [12].

The PCSE Life Cycle

The Practice Centered Software Engineering (PCSE) is the most recent version of Auburn University's personal self-improvement process for helping software engineers control, manage, and improve the way they work. PCSE was developed to introduce a light-weight software process and practice the major software process tools that are widely recognized and used in the industry.

This chapter explains the PCSE life cycle. The different activities involved in PCSE are explained in detail. Each artifact that belongs to a particular activity is explained with sample examples.

All information, content, explanations and examples referred to in this chapter originated from the following sources:

1.) Dr. David A. Umphress personal communication.

2.) "Software Process" (COMP 6700) class presentations, videos.

3.) Interaction with the PCSE research team.

Although PCSE is well defined, it undergoes continuous improvement. There is considerable room for enhancing PCSE by introducing new techniques and tools. The current PCSE research team is working along with Dr. Umphress to identify the scope for improvement.

The following figure illustrates the different activities and their flow involved in PCSE:

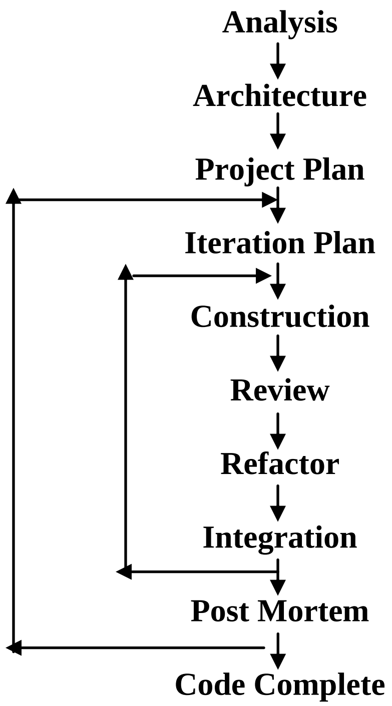


Figure 1: PCSE Life Cycle

Each of the different activities and their corresponding artifacts is described below:

4.1 Analysis:

Analysis is the process of breaking a complex topic into smaller parts to gain a better understanding of it. This stage includes identifying the desired behavior of the system. The outcome of the analysis phase is an operational specification which is a list of scenarios, where each scenario is a representative collection of desired behaviors expected of the software component under consideration [1].

There are two major types of operational specification:

1.) Interface operational specification – illustrates component-to-component interaction

2.) User operational specification – illustrates user-to-component interaction

The following are the examples of Interface and User operational specifications:

Tuple #	Type	Actor	Event/Actor response description	Example
1	Event	Test Driver	call average with valid list	Statistics.average([1,2,3,4,5])
2	Response	Blackbox	returns average of list	3.0
3	Event	Test Driver	call median with valid list containing odd number of values	Statistics.median([1,2,3])
4	Response	Blackbox	returns median of list	2.0
5	Event	Test Driver	call median with valid list containing even number of values	Statistics.median([1,2,3,4])
6	Response	Blackbox	returns median of list	3.5
7	Event	Test Driver	call stdev with valid list	Statistics.stdev(list)
8	Response	Blackbox	returns standard deviation of list	1.29

Table 1: Interface Operational Scenario – nominal [1]

Tuple #	Type	Actor	Event/Actor response description	Example
1	Event	Test Driver	call stdev with empty list	Statistics.stdev([])
2	Response	Blackbox	raises exception	Runtime Error
3	Event	Test Driver	call stdev with one-element list	Statistics.stdev([5])
4	Response	Blackbox	raises exception	Runtime Error

Table 2: Interface Operational Scenario – anomalous [1]

Tuple #	Type	Actor	Event/Actor response description	Example
1	Event	User	Start application	
2	Response	Blackbox	"Enter filename or stop"	
3	Event	User	User enters file name	assignment1test1.txt
4	Response	Blackbox	Display number of values in the file	10
5	Response	Blackbox	Display average of values in the file	42
6	Response	Blackbox	Display the median of the values in file	39.5
7	Response	Blackbox	Display the stdev of the values in the file	2.7
8	Response	Blackbox	"Enter filename or stop"	
9	Event	User	User enters "stop"	
10	Response	Blackbox	"Program terminated"	

Table 3: User Operational Scenario – nominal [1]

4.2 Architecture:

The main motive of this phase is to develop a high-level design and to identify major components sufficient to begin scoping the effort required by the project. This phase focuses on allocating functionality. During this phase, the output of the analysis phase is used to partition the system into conceptual components using CRC (Class Responsibility Collaborator) cards. This activity entails identifying parts within the black box at a limited level of abstraction, i.e., identifying major components called proxies, usually objects and functions. This provides the basis for estimation, task identification and scheduling. CRC cards can be visualized as a textual/tabular version of UML class diagrams.

A CRC card contains the following elements:

a.) **Proxy Name:** denotes the name of the class or function

b.) **Design Approach:** either Object Oriented (or) Functional

c.) **Super class:** denotes a parent proxy

d.) **Component Type:** either Logic (or) Calculation (or) Data (or) Input/output

e.) **Collaborators:** represents other components which has a relationship with this component

f.) **Operations:** represents the functionalities

The following are examples of CRC cards:

1.)	*Proxy Name:*	print_proxy_history
	Design Approach:	Functional
	Parent Proxy:	
	Attributes (optional):	
	Component Type:	I/O
	Collaborators:	ProxyHistory
	Operations:	print_proxy_history

2.)	*Proxy Name:*	ProxyHistory
	Design Approach:	Object-oriented
	Parent Proxy:	
	Attributes (optional):	
	Component Type:	Logic
	Collaborators:	SourceFile

Operations: initialize, generate_project_history, generate_proxy_history calculate_average_size

3.) *Proxy Name:* SourceFile

Design Approach: Object-oriented

Parent Proxy:

Attributes (optional):

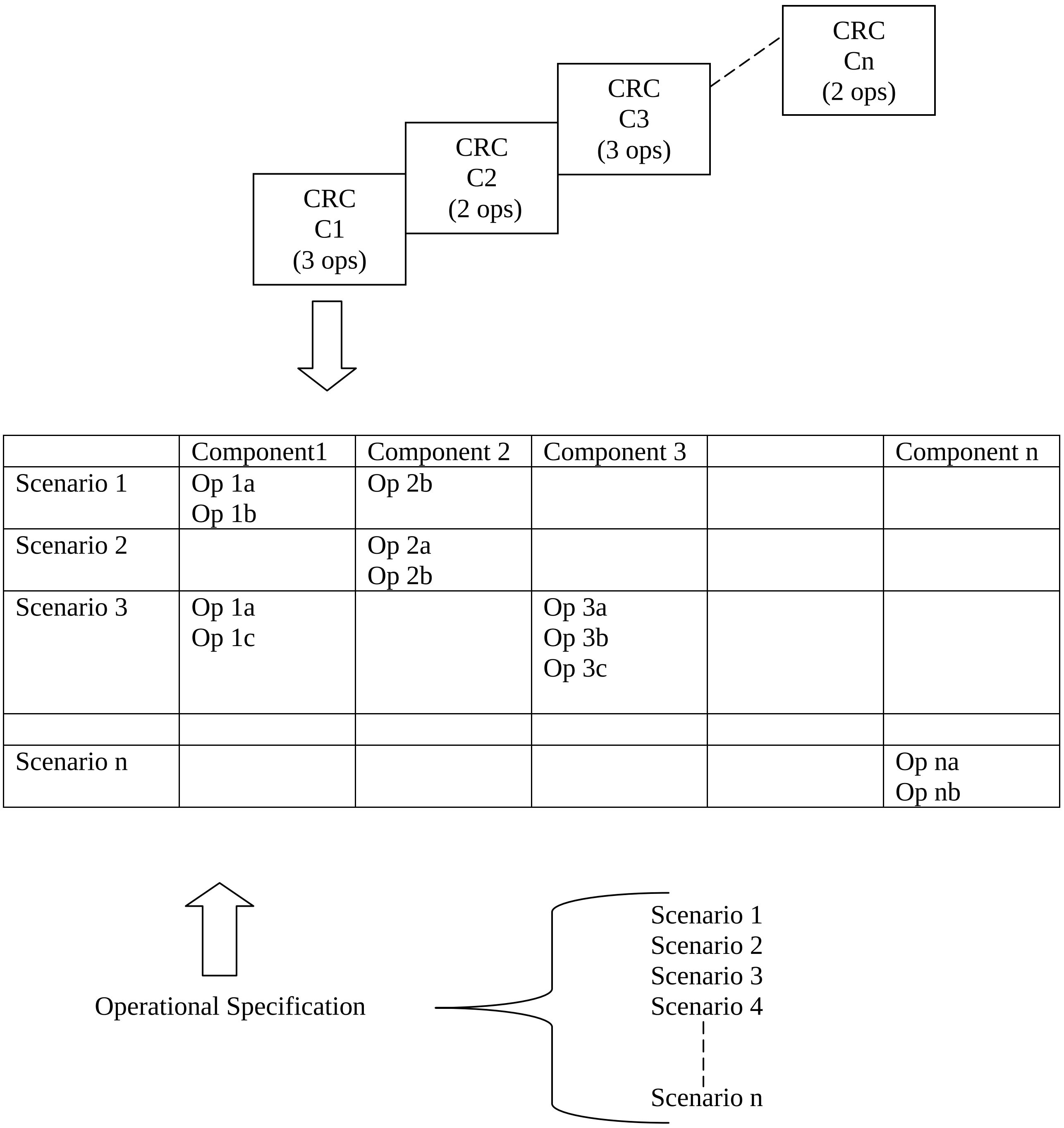
Component Type: Calculation

Collaborators:

Operations: initialize, count_lines, validate_count, validate_blocks, file_name, count_proxyblock, validate_proxyblocks, count_proxymethod, get_proxytype, each_proxy

After identifying the major components (CRC cards) in the Architecture phase and the scenarios in the Analysis phase, the developer comes up with a Scenario-Component Map. This allows the developer to map each operation in the component to their respective scenarios.

The following table illustrates the scenario-component map:



	Component1	Component 2	Component 3		Component n
Scenario 1	Op 1a Op 1b	Op 2b			
Scenario 2		Op 2a Op 2b			
Scenario 3	Op 1a Op 1c		Op 3a Op 3b Op 3c		
Scenario n					Op na Op nb

Figure 2: Scenario – Component map

4.3 Project Plan:

The main motive of this phase is to estimate the overall effort. In PCSE, estimation is done by Proxy-Based Estimation setup, where it relies on historical project data (i.e. both project history and proxy history). The Project and Proxy history consists of the following data:

Project History:

1.) LOCr – Raw lines of code

2.) LOCp – Planned lines of code

3.) LOCa – Actual lines of code

4.) Ep – Planned duration

5.) Ea – Actual duration

Proxy history:

1.) Proxy Name

2.) Total LOC

3.) Methods

4.) Type

5.) Size

6.) LOC/method

7.) Ln(LOC/Method)

First, a size matrix is constructed based on the historical project data, which results in calculating the size ranges. The size ranges are categorized into Very Small (VS), Small (S), Medium (M), Large (L), and Very Large (VL). The size matrix is used to derive the raw lines of code (LOCr) for the new project, from which the planned lines of code (LOCp) and the planned estimate time (Ep) can be calculated.

The following section explains in detail the calculation of the size matrix and deriving the LOCp and Ep for a new project.

Let's assume we have the following project and the proxy history.

Project history:

Project Name	**LOCr**	**LOCp**	**LOCa**	**Ep**	**Ea**
Project 1	30	30	48	145	249
Project 2	45	45	168	190	419
Project 3	65	65	146	290	438
Project 4	273	339	274	627	577
Project 5	203	270	182	589	513

Table 4: Example project history

Proxy history:

Proxy Name	**Total LOC**	**Methods**	**Type**	**LOC/meth**	**ln(LOC/meth)**
SourceFile	48	4	Calculation	12.00	2.48
SourceFile	211	10	Calculation	21.10	3.05
print_each_proxy	20	1	I/O	20.00	3.00
ProxyHistory	120	4	Logic	30.00	3.40
print_proxy_history	25	1	I/O	25.00	3.22
print_schedule	45	1	I/O	45.00	3.81
Schedule	178	3	Calculation	59.33	4.08
TaskList	19	3	Logic	6.33	1.85
Task	12	3	Data	4.00	1.39
Calendar	20	3	Data	6.67	1.90
print_tcurve	43	1	I/O	43.00	3.76
TCurve	139	7	Calculation	19.86	2.99

Table 5: Example proxy history

The Mean and the Standard Deviation are calculated using the following:

Mean = AVERAGE (ln (LOC/meth)**)**

Std Dev = STDEV (ln (LOC/meth))

4.3.1 Calculating the size matrix:

The size matrix is calculated using the following template:

	Low	Mid	High
VS	1	=CEILING(EXP(Average-2*StdDev),1)	=CEILING(EXP(Average-1.5*StdDev),1)
S	=CEILING(EXP(Average-1.5*StdDev),1)	=CEILING(EXP(Average-StdDev),1)	=CEILING(EXP(Average-0.5*StdDev),1)
M	=CEILING(EXP(Average-0.5*StdDev),1)	=CEILING(EXP(Average), 1)	=CEILING(EXP(Average+0.5*StdDev),1)
L	=CEILING(EXP(Average+0.5*StdDev),1)	=CEILING(EXP(Average+StdDev),1)	=CEILING(EXP(Average+1.5*StdDev),1)
VL	=CEILING(EXP(Average+1.5*StdDev),1)	=CEILING(EXP(Average+2*StdDev),1)	big

Table 6: Size matrix formula

The size matrix can be calculated for each type (i.e. Calculation, I/O, Logic, and Data) (or) without considering the types. The following is the size matrix calculation based on the example proxy history provided in Table 5.

Proxy Name	Type	LOC/meth	ln(LOC/meth)
SourceFile	Calculation	12.00	2.48
SourceFile	Calculation	21.10	3.05
print_each_proxy	I/O	20.00	3.00
ProxyHistory	Logic	30.00	3.40
print_proxy_history	I/O	25.00	3.22
print_schedule	I/O	45.00	3.81
Schedule	Calculation	59.33	4.08
TaskList	Logic	6.33	1.85
Task	Data	4.00	1.39
Calendar	Data	6.67	1.90
print_tcurve	I/O	43.00	3.76
TCurve	Calculation	19.86	2.99

	Low	Mid	Upper
VS	1	3	5
S	5	8	12
M	12	18	28
L	28	43	66
VL	66	100	big

Table 7: Size matrix calculation without considering type.

Once the size matrix is calculated, the raw lines of code (LOCr) for the new project can be calculated by the following steps:

1.) Identify the relative size for each of the history proxies from the size matrix. This is done by getting the LOC/method for each historical proxy and finding which bucket it falls into. For example, the proxy "ProxyHistory" in the Table 5 has a LOC/Method of 30. If we map 30 in the size matrix, it falls in the Large (L) category.

The following table shows the relative size for each of the history proxies by mapping it with the size matrix:

Proxy Name	**Type**	**LOC/meth**	**ln(LOC/meth)**	**Rel Size**
SourceFile	Calculation	12.00	2.48	**M**
SourceFile	Calculation	21.10	3.05	**M**
print_each_proxy	I/O	20.00	3.00	**M**
ProxyHistory	Logic	30.00	3.40	**L**
print_proxy_history	I/O	25.00	3.22	**M**
print_schedule	I/O	45.00	3.81	**L**
Schedule	Calculation	59.33	4.08	**L**
TaskList	Logic	6.33	1.85	**S**
Task	Data	4.00	1.39	**VS**
Calendar	Data	6.67	1.90	**S**
print_tcurve	I/O	43.00	3.76	**L**
TCurve	Calculation	19.86	2.99	**M**

Table 8: Relative size mapping from size matrix.

2.) In order to calculate the raw lines of code (LOCr) for the new project, list down all the proxies and the number of operations from the components identified in the Architecture phase.

3.) Identify the estimated relative size (i.e. VS, S, M, L, or VL) from the size matrix for each of the listed proxies in the new project by comparing it with the historical proxy.

4.) The raw lines of code (LOCr) for each proxy in the new project are calculated by multiplying the number of operations/methods with the relative size (Mid value) from the size matrix.

4.3.2 Example for calculating raw lines of code (LOCr):

Let's assume we have the following new proxies in our new project:

Proxy Name	**Operations**
print_checkconsistency	1
check_consistency	1
Cvalidate	8

Table 9: New proxies

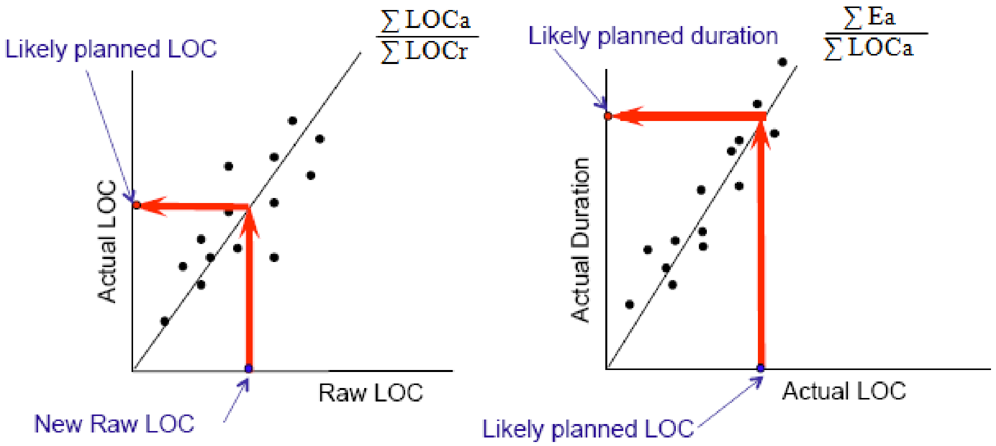
By comparing the historical proxies, we identify the estimated relative size for each of these proxies from the size matrix. Then, the mid value of the corresponding relative size from the size matrix is taken and multiplied with the number of operations to get the LOCr. Finally the total LOCr is calculated by summing up the LOCr of all the new proxies as shown below:

Proxy Name	Operations	Estimated Rel. Size	LOCr
print_checkconsistency	1	**M**	**18**
check_consistency	1	**L**	**43**
Cvalidate	8	**L**	**344**
			Total LOCr = 405

Table 10: Raw lines of code (LOCr) calculation for new proxies

4.3.3 Calculating Planned lines of code (LOCp) and Planned duration (Ep):

Once the raw lines of code (LOCr) is calculated, we can calculate the planned lines of code (LOCp) and the planned duration(Ep) with the help of the project history. The following estimation fundamentals will help us understand better how LOCp and Ep are calculated:



$$\mathit{New}\ \mathrm{LOCr} * \frac{\sum \mathrm{LOCa}}{\sum \mathrm{LOCr}} = \mathit{New}\ \mathrm{LOCp}$$

$$\mathit{New}\ \mathrm{LOCp} * \frac{\sum \mathrm{Ea}}{\sum \mathrm{LOCa}} = \mathit{New}\ \mathrm{Ep}$$

Figure 3: Estimation Fundamentals

In the left graph, the dots represent the number of historical projects. The x-axis represents the raw lines of code (LOCr) and the y-axis represents the actual lines of code (LOCa). The line that passes through the historical projects represents the average of LOCa to LOCr. The blue dot represents the raw lines of code (LOCr) for the new project/development (i.e. 405 lines in our example case). The red line that passes through the line is a mapping of LOCr to LOCa for the new development, which results in the planned lines of code (LOCp) for the new development (i.e. *New* LOCp is obtained by multiplying *New* LOCr with the average). The average ∑ LOCa / ∑ LOCr actually represents the productivity. For example, if the average is 1.33, it means that 1.33 lines of code can be written in one minute, but again it depends upon the language, code etc.

In the right graph, the dots represent the number of historical projects. The x-axis represents the actual lines of code (LOCa) and the y-axis represents the actual duration (Ea). The line that passes through the historical projects represents the average of Ea to LOCa. The blue dot represents the planned lines of code (LOCp) for the new project/development. The red line that passes through the line is a mapping of LOCa to Ea for the new development, which results in the planned duration (Ep) for the new development (i.e. *New* Ep is obtained by multiplying *New* LOCp with the average). The average ∑ Ea / ∑ LOCa actually represents the productivity. For example, if the average is 2.68, it means that it almost take 3 minutes to write a line of code, but again it depends upon the language, code etc.

Let’s assume from the historical database, we have the following data:

Project Name	**LOCr**	**LOCa**	**Ea**
Project 1	30	48	249
Project 2	45	168	419
Project 3	65	146	438
Project 4	273	274	577
Project 5	203	182	513
$\sum \text{LOCa} / \sum \text{LOCr} = 1.33$ $\sum \text{Ea} / \sum \text{LOCa} = 2.68$			

Table 11: Historical project data

Therefore, If the raw size (LOCa) is 405, then the planned size (LOCp) is ceil(405 * 1.33) = **539 lines** and the planned duration (Ep) is ceil(539 * 2.68) = **1445 minutes.**

4.3.4 Calculating confidence:

Calculating confidence involves calculating lower prediction interval (LPI) and upper prediction interval (UPI). The rule is as follows:

For size estimates:

LPI = Estimate * largest historical underestimation error

UPI = Estimate * largest historical overestimation error

For time estimates:

LPI = Estimate * fastest historical productivity

UPI = Estimate * slowest historical productivity

From the historical data, we can calculate the size prediction interval and time prediction interval as follows:

Project Name	**LOCr**	**LOCa**	**Ea**	**LOCa/LOCr**	**(LOCa/Ea) * 60** (Unit: LOC/Hr)
Project 1	30	48	249	1.6	11.4
Project 2	45	168	419	3.73	24.0
Project 3	65	146	438	2.25	19.8
Project 4	273	274	577	1.00	28.8
Project 5	203	182	513	0.90	21.0

New LOCr = 405; *New* LOCp = 539

Table 12: Size and Time prediction intervals calculation

4.3.5 Size prediction interval:

Therefore, Lower Prediction Interval **(LPI)** = biggest underestimation error

= New LOCr * min(LOCa/LOCr)

= 405 * 0.90

= **365 lines**

Upper Prediction Interval **(UPI)** = biggest overestimation error

= New LOCr * max(LOCa/LOCr)

= 405 * 3.73

= **1511 lines**

4.3.6 Correlation coefficient:

The correlation coefficient, a concept from statistics is a measure of how well trends in the predicted values follow trends in past actual values. It is a measure of how well the predicted values from a forecast model "fit" with the real-life data.

The correlation coefficient is a number between 0 and 1. If there is no relationship between the predicted values and the actual values, the correlation coefficient is 0 or very low (the predicted values are no better than random numbers). As the strength of the relationship between the predicted values and actual values increases so does the correlation coefficient. A perfect fit gives a coefficient of 1.0. Thus the higher the correlation coefficient the better [13].

Let the correlation coefficient ranges be set as follows:

$1.0 < r^2 < .75$ --------- High (means estimation very close to actual values)

$.75 < r^2 <.50$ ---------- Medium (means estimation fairly close to actual values)

$.50 < r^2$ ---------------- Low (means estimation is very poor)

4.3.7 Confidence on size prediction interval:

In order to find the confidence on the size prediction interval, calculate correlation coefficient which is done as follows:

= [CORREL(∑ LOCa / ∑ LOCr)]^2

= **0.70 [Medium]**

Note:

set UPI to *New* LOCp if UPI< *New* LOCp

set LPI to 1 if LPI > *New* LOCp

also set confidence to low

4.3.8 Time prediction interval:

Therefore, Lower Prediction Interval (**LPI**) = biggest overestimation error

= *New* LOCp / max(LOCa/Ea) * 60

= 539 / 28.8 * 60

= **1123 minutes**

Upper Prediction Interval (**UPI**) = biggest underestimation error

= *New* LOCp / min(LOCa/Ea) * 60

= 539 / 11.4 * 60

= **2837 minutes**

4.3.9 Confidence on time prediction interval:

In order to find the confidence on the time prediction interval, calculate correlation coefficient which is done as follows:

= [CORREL(∑ Ea / ∑ LOCa)]^2

= **0.93 [High]**

Note:

set UPI to *New* Ep if UPI< *New* Ep

set LPI to 1 if LPI > *New* Ep

also set confidence to low

4.4 Iteration Plan:

IBM Rational Unified Process (RUP) [14], an iterative software development process framework says that iteration plan is a fine-grained plan with a time-sequenced set of activities and tasks, with assigned resources, containing task dependencies, for the iteration.

Iteration also involves the redesign and implementation of a task from the operational specification list, and the analysis of the current version of the system. It helps identify problems or faulty assumptions at periodic intervals.

PCSE has the following major goals during this phase:

a) Select/revise scenario set

b) Set iteration goal

c) Schedule work

4.4.1 Select/revise scenario set:

During every iteration, one can select the next set of scenarios identified in the Analysis phase, for the implementation. The operational specification can also be revisited to modify any scenario or add new scenarios.

4.4.2 Set iteration goal:

The iteration goal is a plan or a set of tasks intended to be achieved by the end of the iteration. Some of the iteration goals are:

1) A list of major classes or packages that must be completely implemented.
2) A list of scenarios or use cases that must be completed by the end of the iteration.
3) A list of risks that must be addressed by the end of the iteration.
4) A list of changes that must be incorporated in the product (bug fixes, changes in requirements) etc.

4.4.3 Schedule work:

One of the important artifacts of iteration plan in PCSE is the iteration map, where each part in the component identified during the architecture phase is mapped to an iteration. However the number of iterations is not limited.

There are situations where we write mock code before we write production code. Each of these can be easily captured and tracked using an iteration map.

The iteration map also leads an easy way to schedule and track the effort, using calendar, burn-down chart, and diary. In the other end, the iteration map can also be mapped to scenarios.

The following table illustrates how each parts (methods) of a component are mapped to each iteration in the iteration map:

	I1	I2	I3	I4	
Comp1					
Op1	Prod				
Op2	Prod				
Op3	Mock	Prod			
Comp2					
Op1		Prod			
Op2		Prod			
Comp3					
Op1		Prod			
Op2			Prod		
Op3		Mock	Mock	Prod	
Comp4					
Op1				Prod	
Op2				Prod	
Total Parts	3	5	2	3	13
Effort	3/13 * 1445 = 333	5/13 * 1445 = 557	2/13 * 1445 = 222	3/13 * 1445 = 333	1445

Table 13: Iteration Map

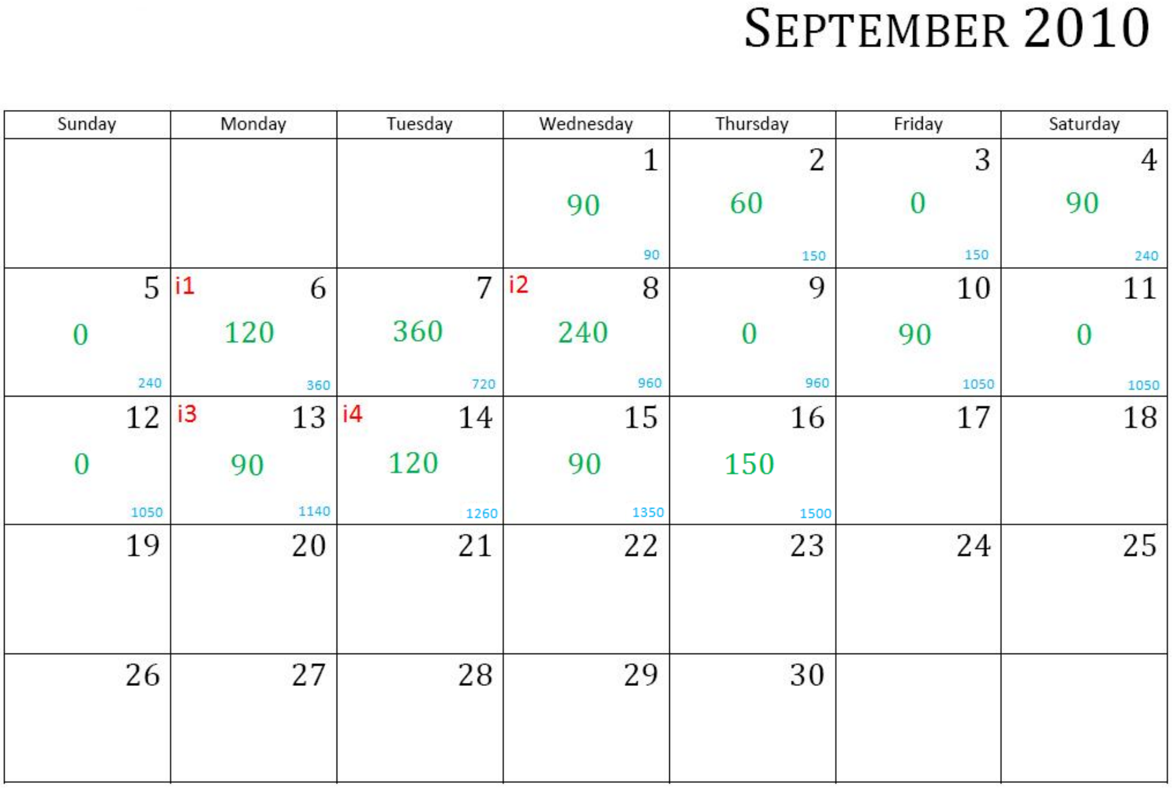
Total parts: 13, Planned effort (Ep) in our case is 1445 minutes

Please Note: 1.) No restrictions on the number of iterations, 2.) assuming each part averages to same effort

In the above table, each operation of a component (CRC card) is mapped to an iteration. The total parts for each iteration can be calculated by counting the number of parts implemented (mock or prod) during that iteration, and the effort for each iteration is calculated by dividing with the total parts and then multiplying with the planned estimate (Ep) (1445 minutes in our example case).

Now that we know the effort for each iteration, the work can be scheduled using a calendar and tracked using the burn down chart and diary. The calendar, burn-down chart, and dairy are important artifacts in PCSE in order to schedule and track the effort.

The following figure illustrates how calendar is used to measure and track effort for every iteration.



SEPTEMBER 2010

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
			1 90 90	2 60 150	3 0 150	4 90 240
5 0 240	i1 6 120 360	7 360 720	i2 8 240 960	9 0 960	10 90 1050	11 0 1050
12 0 1050	i3 13 90 1140	i4 14 120 1260	15 90 1350	16 150 1500	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Figure 4: Calendar

The iteration map is scheduled to complete 13 parts in 4 iterations with an estimated duration of 1445 minutes. The calendar is used to plan the effort on a daily basis. In the above figure, the numbers in the center (green in color) represents the number of minutes the developer has planned on writing the part on the given day. The number at the bottom right end (blue in color) of each day is the cummulative total of minutes planned so far (cummulative planned time), from which the day an iteration will end is known). For example, iteration1 (i1) requires an effort of 333 minutes. So, from the calendar above, iteration1 will be completed on day 6. The end of each iteration is marked in red at the top left end of the completion day. The actual effort spent during each day can be tracked using a burn-down chart and a diary.

4.4.4 Burn-down chart:

A burn-down chart is a graphical representation of work left to do versus time. The outstanding work (or backlog) is often on the vertical axis, with time along the horizontal. That is, it is a run chart of outstanding work. It is useful for predicting when all of the work will be completed. The end of each day is termed as the recording point and the end of each iteration is termed as assessment point [15].

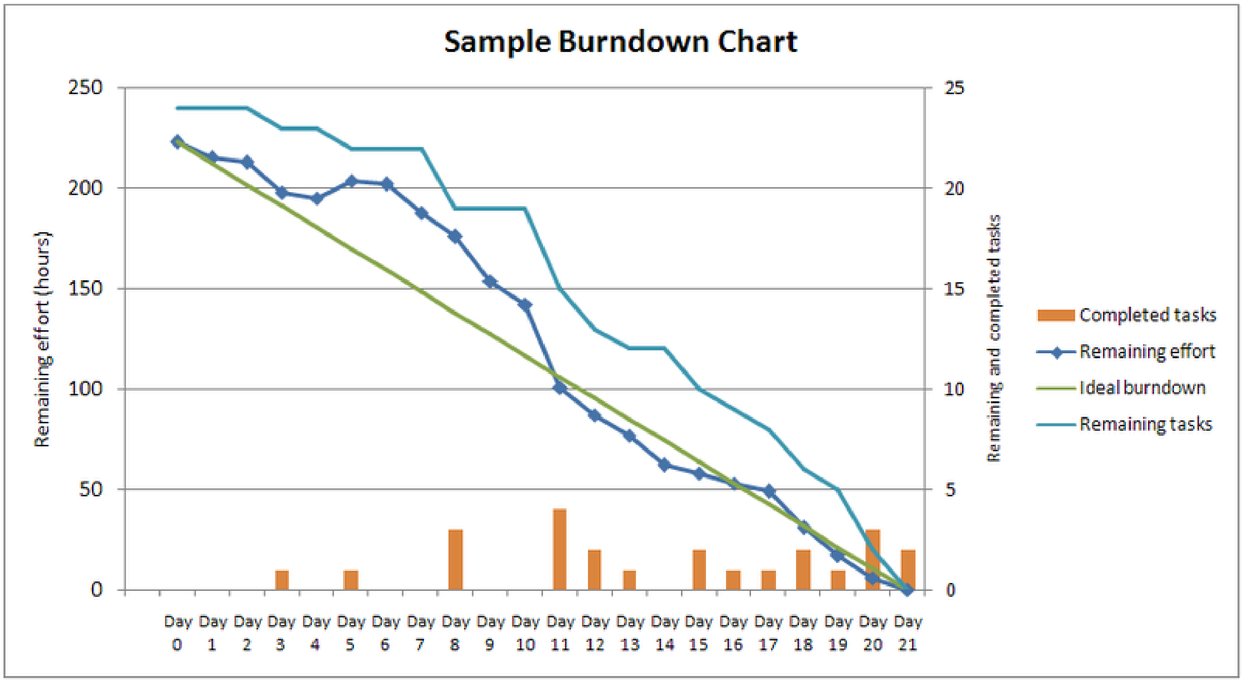


Figure 5: A sample burn-down chart for a completed iteration, showing remaining effort and tasks for each of the 21 work days of the 1-month iteration.

4.4.5 Burn down chart example case:

The following is an example burn-down chart constructed based on the work scheduled in the iteration map and the calendar. It represents the Planned Effort(Ep), Actual Effort (Ea) and the replanned effort. i1, i2, i3, i4 etc. represent each iteration. The x-axis represents the days and the y-axis refers to remaining effort.

After each day the progress is marked on the chart and after each iteration the developer projects forward to see whether or not the target end date will be hit.

In the diagram below, the developer had gone through three iterations, and the dotted line (work effort to maintain schedule) suggests he had fallen quite badly behind schedule. After the first iteration the developer was making good progress, but things changed in the second and third iteration. Although the developer was behind the planned schedule at the end of the second iteration, he thought that he could still finish the remaining parts. Since the remaining time was not realistic to finish the remaining parts, the developer decided to reschedule at the end of third iteration.

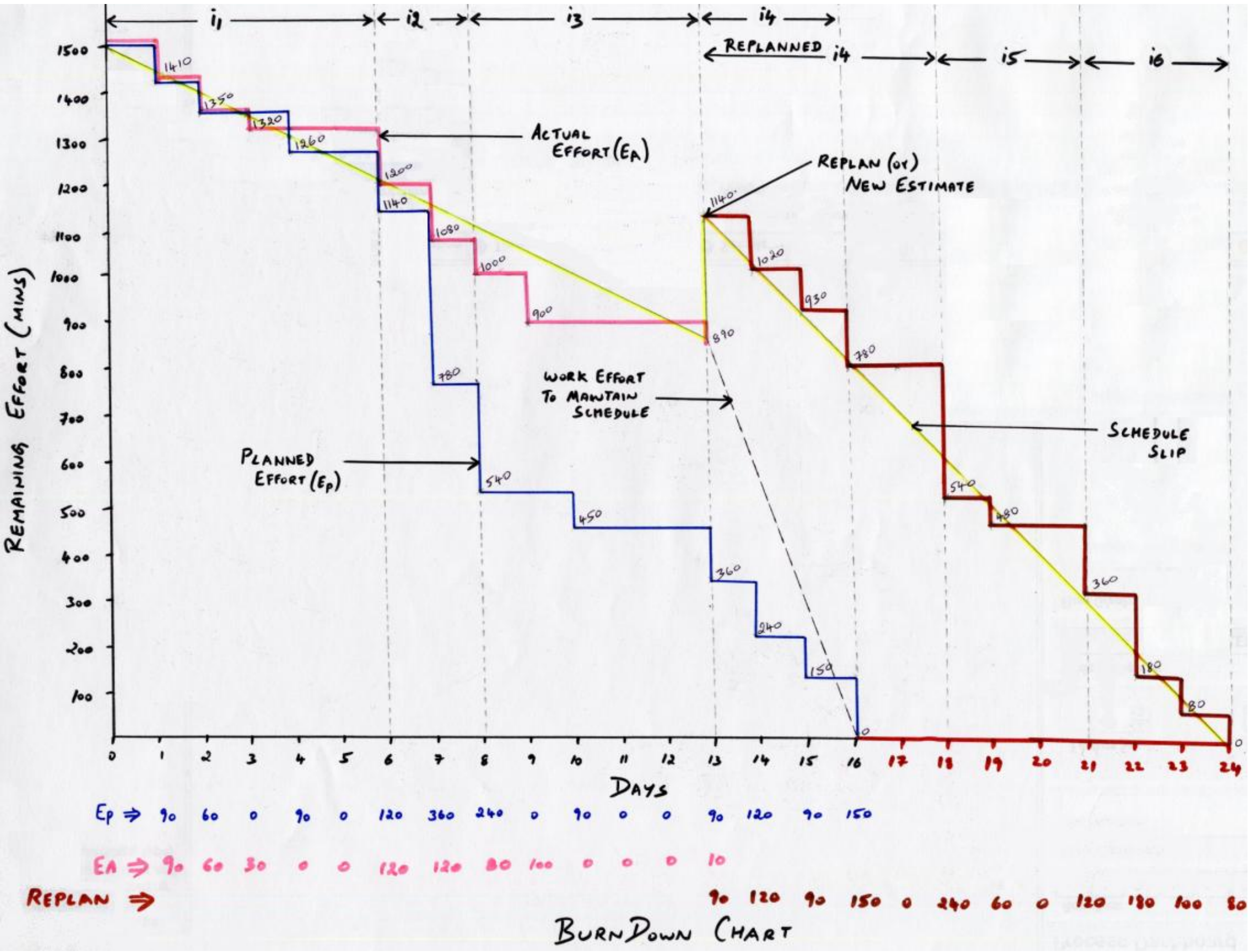


Figure 6: Example burn-down chart

4.4.6 Diary:

A diary is used for measuring the project progress. It provides a way to track and make decisions at periodic intervals, particularly when the developer completes the tasks in a different order than originally planned. Planned Value (PV) and Earned Value (EV) are two important measures that represent the planned work and completed work respectively. PV and EV are also termed as Planned Velocity and Earned Velocity.

Planned Value (PV): The percentage of total planned project time that the planned task represents.

Earned Value (EV): The planned value of a task is earned when the task is completed. There is no partial credit for partially completed tasks; i.e. Earning a EV means method is completed passing all the tests.

If the Earned value (EV) is less than the Planned value (PV), then the developer is accomplishing the work late or behind schedule. If the Earned value (EV) is equal or more than the Planned value (PV), then the developer is accomplishing the work on, or ahead of plan.

A diary holds both the planned and the actual data. The following are the important data used in the Diary for measuring the progress:

Planned Time: Planned effort for a day.

Planned Burn Down: Total planned effort – Planned Time.

Planned Iteration: Mark the completion of a set of planned parts.

Cum PV: The running cumulative sum of the planned values.

Actual Time: Actual effort spent on planned tasks on a day.

Cum Actual Time: The running cumulative sum of the actual time.

Actual Burn Down: Total actual effort – Actual time.

Cum EV: The running cumulative sum of earned values for the completed tasks.

Backlog Δ***:*** represent the total amount of work.

The following is an example diary constructed based on the work scheduled in the iteration map and the calendar. From the historical project data, the estimated planned duration (Ep) was calculated as 1445 minutes. The number of parts determined in the iteration plan was

13 parts in 4 iterations (i1: 3 parts, i2: 5 parts, i3: 2 parts, i4: 3 parts). The number of days scheduled in the calendar was 16 days.

After each day the progress is marked on the chart and after each iteration the developer project forward to see whether or not he will hit the target end date. At the end of iteration 1 (i1) on day 6, the developer had completed 3 parts and so the earned value (EV) is 3. In, iteration 2 (i2), the developer did not progress well and so he earned only 1 part as against the planned value (PV), 5 parts. Although the developer was behind the planned schedule at the end of the second iteration, he thought that he could still finish the remaining parts. At the end of iteration 3 (i3) on day 13, the developer had earned only 1 part against the PV, 3 parts. Also, in iteration 3 (i3), the developer discovered that he need to write 2 more new parts in order to complete the project. Since the remaining time was not realistic to finish the remaining parts, the developer decided to reschedule at the end of third iteration.

Day	Planned Time	Planned Burn Down	Planned Iteration	Cum PV	Actual Time	Cum Actual Time	Actual Burn Down	Cum EV	Backlog Δ
1	90	1410		0	90	90	1410	0	0
2	60	1350		0	60	150	1350	0	0
3	0	1350		0	30	180	1320	0	0
4	90	1260		0	0	180	1320	0	0
5	0	1260		0	0	180	1320	0	0
6	120	1140	i1: 1167 (Effort: 333)	3	120	300	1200	3	-3
7	360	780		3	120	420	1080	3	0
8	240	540	i2: 610 (Effort: 557)	8	80	500	1000	4	-1
9	0	540		8	100	600	900	5	-1
10	90	450		8	0	600	900	5	0
11	0	450		8	0	600	900	5	0
12	0	450		8	0	600	900	5	0
13	90	360	i3: 388 (Effort: 222)	10	10	610	890	5	+2
14	120	240		10					
15	90	150		10					
16	150	0	i4: 55 (Effort: 333)	13					

Table 14: Diary (Example case) – Planned Vs Actual

At the end of i3 on day 13, the developer earned 5 parts and also discovered that he need to write 2 more new parts. Hence the total amount of work (Backlog) to complete becomes:

Backlog of work = 13 (planned parts) – 5 (EV at i3)

= 8 + 2 (newly discovered)

= 10 parts

The total backlog is 10 parts and the developer decided to change the iteration plan such that he will complete the remaining parts in 3 iterations as follows: (i4: 4 parts, i5: 3 parts, i6: 3 parts). To re-estimate the time for each iteration, the developer has to make use of the cummulative actual time spent and number of EV earned until end of i3. The calculation is done as follows:

Re-estimate (i4) = cum actual time / items built (EV) * items to build in i4

= 610 / 5 * 4

= 488

Re-estimate (i5) = cum actual time / items built (EV) * items to build in i5

= 610 / 5 * 3

= 366

Re-estimate (i5) = cum actual time / items built (EV) * items to build in i6

= 610 / 5 * 3

= 366

The total new planned duration is: 1220 (488 + 366 + 366).

The developer starts recalculating the burndown at the end of i3 on day 13 as follows:

Recalculate burndown = (new planned duration - time remaining in the day (Day 13))

= 1220 – 80

= 1140

Once the burndown is recalculated, the calendar is scheduled and the planned time for each day is recorded in the diary. The following is the diary after re-estimate.

Day	Planned Time	Planned Burn Down	Planned Iteration	Cum PV	Actual Time	Cum Actual Time	Actual Burn Down	Cum EV	Backlog Δ
1	90	1410		0	90	90	1410	0	0
2	60	1350		0	60	150	1350	0	0
3	0	1350		0	30	180	1320	0	0
4	90	1260		0	0	180	1320	0	0
5	0	1260		0	0	180	1320	0	0
6	120	1140	i1: 1167 (Effort: 333)	3	120	300	1200	3	-3
7	360	780		3	120	420	1080	3	0
8	240	540	i2: 610 (Effort: 557)	8	80	500	1000	4	-1
9	0	540		8	100	600	900	5	-1
10	90	450		8	0	600	900	5	0
11	0	450		8	0	600	900	5	0
12	0	450		8	0	600	900	5	0
13	90	**1140**	i3: 388 (Effort: 222)	10	10	610	890	5	+2
recalculate burndown (1220 - time remaining in the day) = 1220 – 80 = 1140									
14	**120**	**1020**		**5**					
15	**90**	**930**		**5**					
16	**150**	**780**		**5**					

17	**0**	**780**		**5**					
18	**240**	**540**	**i4: 732** **(Effort: 488)**	**9**					
19	**60**	**480**		**9**					
20	**0**	**480**		**9**					
21	**120**	**360**	**i4: 366** **(Effort: 366)**	**12**					
22	**180**	**180**		**12**					
23	**100**	**80**		**12**					
24	**80**	**0**	**i4: 0** **(Effort: 366)**	**15**					

Table 15: Diary (Example case) after Re-estimate

4.5 Construction:

Construction is the activity of building code given a high-level design, which involves low-level design, coding and unit testing.

In PCSE, the following are the important activities of this phase:

a.) Develop low-level design

b.) Build code and unit tests via TDD

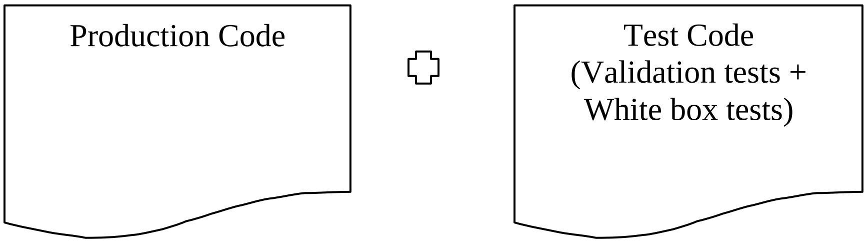


Figure 7: PCSE construction artifacts

Traditionally testing was done after the coding activity, but

PCSE uses Test Driven Development (TDD) approach where first the tests are written and then code to pass them.

Design → Code → Test
Traditional Approach

Design → Test → Code
PCSE Approach

The problems of the traditional approach are:

a.) Tests are often written based on non-code artifacts, which may work for black-box tests, but not necessarily for white-box tests

b.) Tests may be written by non-coders who lack white box knowledge.

c.) Testing is done after code is written

4.5.1 Test Driven Development [16]:

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes a failing automated test case that defines a desired improvement or new function, and then produces code to pass that test [16]. It is a systematic approach to programming where the tests determine what code to write.

The advantages of TDD approach are:

a.) all delivered code is accompanied by tests

b.) no code goes into production untested

c.) writing tests first yield a better understanding of what the code is to do

The following figure represents a complete test driven development cycle:

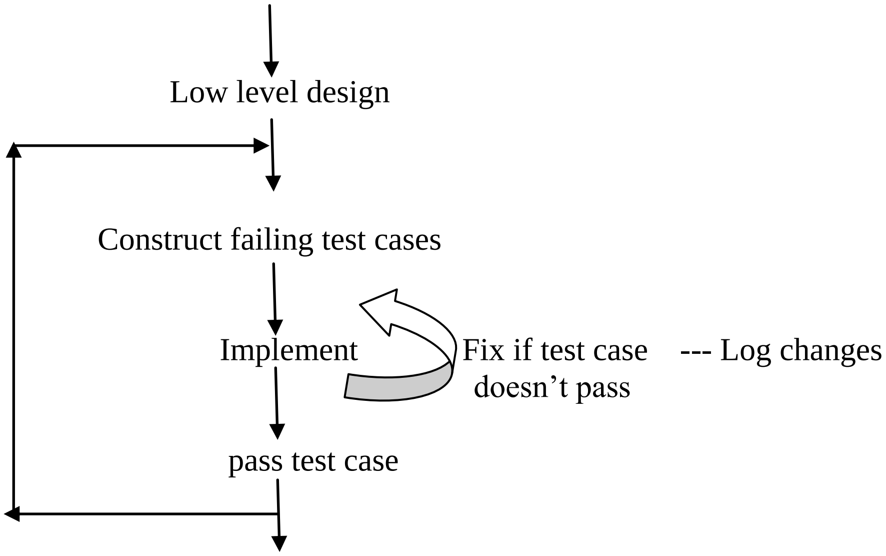


Figure 8: Test driven development cycle

Test cases can also be written using automated testing tools such as JUnit, NUnit etc.

4.6 Review:

Review is a phase where the developer examines the test code for coverage. This is the phase where the developer ensures enough black box tests and white box tests are constructed. If the test code is not covered, then the developer can add tests appropriately and fix it via TDD.

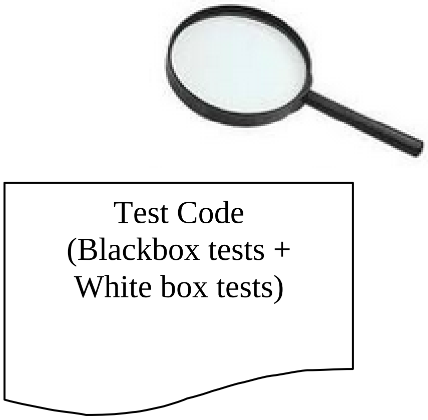


Figure 9: Review Test Code

The better the quality, the lower the cost. Test code reviews help ensure that quality and standards. It will not only improve the quality of the test code, but it can also expose any assumptions made by the developer and help in generating more test cases. Some of the most common problems to look for during a test code review are:

a) Creating test code that tests functionality already tested in a different test method. For example testing a "get" along with a “set” and also having a separate test method for testing the "get".

b) Testing the same functionality dropped in two different releases by rewriting existing test cases just for the new release. Existing test code should be written in a way that it works across multiple releases.

c) Re-writing functions that have part of the same functionality implemented by other methods should be avoided. For example, if functionality involves opening a file, reading a value from it, connecting to a database and getting values out and if methods exist for each of these tasks then they should be reused instead of creating a new method that does all three tasks.

d) Verifications should be performed for all possible fields of the object being tested. This should include core fields and also audit fields such as created, updated date etc. [17]

The advantages of performing a test code review are:

a) Tests reveal the intention behind the code much better than the code itself. That means it's easier to discover logical bugs in the code by reading a test.

b) Tests are declarative by default - the developer declares what the code is supposed to be accomplishing.

c) Tests are faster and shorter to read and understand. [18]

4.7 Refactor:

Code refactoring is the process of changing a computer program's source code without modifying its external functional behavior in order to improve some of the nonfunctional attributes of the software. Also the internal structure of the software is improved. Advantages include improved code readability and reduced complexity to improve the maintainability of the source code, as well as a more expressive internal architecture or object model to improve extensibility [19].

The purpose of refactoring is

a.) To make software easier to understand

b.) To help find bugs

c.) To prepare software for next iteration

d.) To speed development process

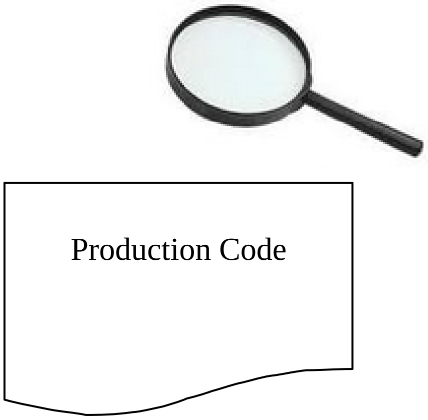


Figure 10: Refactor production code

Refactoring is usually motivated by noticing a code smell. For example the method at hand may be very long, or it may be a near duplicate of another nearby method. Once recognized, such problems can be addressed by refactoring the source code, or transforming it into a new form that behaves the same as before but that no longer "smells". We can also add/modify tests appropriately and fix it via TDD, if we notice code smell in the production code.

The following are some examples of code smell that can be addressed through refactoring [1]:

a.) ***Data clumps:*** member fields that clump together but are not part of the same class.

b.) ***Primitive obsession:*** characterized by the use of primitives in place of class methods

c.) ***Switch statements:*** often duplicated code that can be replaced by polymorphism

d.) ***Parallel inheritance hierarchies:*** duplicated code in subclasses that share a common ancestor.

e.) ***Duplicated code***

f.) ***Long method***

g.) ***Large class***

h.) ***Long parameter list***

i.) ***Divergent change:*** one type of change requires changing one subset of modules; another type of change requires changing another subset.

j.) ***Shotgun surgery:*** a change requires a lot of little changes to a lot of different classes.

k.) ***Feature envy:*** a method in a class seems not to belong.

l.) ***Lazy class:*** a class that has little meaning in the context of the software

m.) ***Speculative generality:*** methods (often stubs) that are placeholders for future features.

n.) ***Temporary field:*** a variable that is used only under certain circumstances and is reused later under other circumstances.

o.) ***Message chains:*** object that requests an object for another object.

p.) ***Middle man:*** a class that is just a "pass-through" method with little logic

q.) ***Inappropriate intimacy:*** violation of private parts.

r.) ***Alternate class with different interfaces:*** two methods that do the same thing, but have different interfaces.

s.) ***Incomplete library classes:*** a framework that doesn't do everything you need.

t.) ***Data class:*** classes that have getters and setters, but no real function.

u.) ***Refused bequest:*** a subclass that over-rides most of the functionality provided by its super class.

v.) ***Comments:*** text that explains bad code (vs fixing the code).

4.8 Integration:

This is a phase where the software component undergoes regression tests before its actual integration.

Regression testing is any type of software testing that seeks to uncover software errors by partially retesting a modified program. The intent of regression testing is to provide a general assurance that no additional errors were introduced in the process of fixing other problems. Regression testing is commonly used to test the system efficiently by systematically selecting the appropriate minimum suite of tests needed to adequately cover the affected change. Common methods of regression testing include rerunning previously run tests and checking whether previously fixed faults have re-emerged. "One of the main reasons for regression testing is that it's often extremely difficult for a programmer to figure out how a change in one part of the software will echo in other parts of the software" [20].

The following methodology is followed in PCSE to ensure all test cases introduced in one particular iteration do not alter the behavior of the overall system:

```
For  i in regression tests
        If not passed(i)
        Declare i invalid & discard
        Defer i to backlog
        Fix i this iteration
```

4.9 Post Mortem:

This is a phase to prepare for the next iteration. The major activities addressed in PCSE during this phase are:

a.) Baseline the production source code in version control

b.) Baseline the test code in version control

c.) Revisit estimation if necessary

d.) Revisit iteration map if necessary

e.) Revisit backlog to add/remove scenarios etc.

4.10 Code Complete:

Code complete is to mark the end of the development. Some of the major activities carried out during this phase are:

1.) Final testing of the system.

2.) User documentation.

3.) Deployment.

4.) Training.

5.) Documentation of findings, information to improve future efforts.

4.11 PCSE Measures:

PCSE has two measures [21]:

1.) The time spent per phase (Timelog).

2.) The defects found per phase (Changelog).

The time per phase is a simple record of the clock time spent in each part of the process. The defects found per phase are a simple record of the specified data for every defect the developer find during compiling and testing.

The reason for gathering both time and defect data will help the developer assess the quality of work and to help plan and manage his projects. These data will show where the developer spends his time and where he inject and fix the most defects. The data will also help the developer see how his performance changes as he modifies the process. The developer can then decide for himself how each process change affects his productivity and the quality of work products.

4.11.1 Timelog:

Timelog is an important artifact used to record the time the developer spends on each project activity.

The following are the important timelog data and their instructions:

General: a.) Record all of the time spend on the project

b.) Record the time in minutes

c.) Be as accurate as possible

d.) If the developer forgets to record the starting, stopping, or interruption time for an activity, promptly enter the best estimate.

Date: Enter the date when the developer start working on a process activity.

Start time: Enter the time when the developer start working on a process activity.

Stop time: Enter the time when the developer stop working on a process activity.

Interrupt time: Record any interruption time (e.g. phone calls, questions or other things) that was spent on the process activity. The reason for the interrupt is usually recorded in the comments section.

Delta: Enter the clock time the developer actually spent working on the process activity, less the interruption time.

Phase: Enter the process activity (e.g. Analysis, Architecture, Project Plan, Iteration Plan etc.).

Feature Set: Enter the current iteration the developer is in.

Comments: Enter any other pertinent comments that might later remind the developer of any unusual circumstances regarding this activity.

The timelog is useful for the following:

1) Calculating the actual time

2) Evaluating what percentage of the developer's time is spent on each area.

3) Determining how much time do the developer usual tasks actually take.

4) Measuring productivity

5) Understanding the interruption problem and figure out ways to address it.

The following is an example timelog:

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Comments
11/2/2008	11:45 PM	11:59 PM		14	Planning	Planned the Historical Data
11/3/2008	12:00 AM	12:15 AM		15	Planning	Planned the Historical Data
11/3/2008	12:25 AM	12:52 AM		27	Planning	Size Matrix
11/3/2008	1:10 AM	1:22 AM		12	Planning	Conceptual Design
11/3/2008	1:22 AM	1:25 AM		3	Planning	Worked on the Planned tab
11/3/2008	1:37 AM	2:07 AM	4	26	Planning	Worked on the Size Estimate tab
11/3/2008	2:10 AM	2:19 AM		9	Planning	Worked on the Schedule tab
11/5/2008	3:19 PM	3:38 PM		19	Design	Worked out the logic and flow for different Classes
11/5/2008	3:40 PM	3:57 PM		17	Design	Designed indepth each loop in the TCurve class
11/5/2008	3:59 PM	4:05 PM		6	Design Review	Reviewed the design
11/5/2008	4:07 PM	4:23 PM		16	Code	Coded the initialize and p methods
11/5/2008	4:24 PM	4:30 PM		6	Code Review	Reviewed the code
11/5/2008	4:35 PM	4:40 PM		5	Test	syntax error, unexpected $end, expecting kEND
11/5/2008	4:42 PM	4:51 PM		9	Test	used abs(n) instead of n.abs
11/5/2008	4:53 PM	4:57 PM		4	Test	Syntax error - usage of)) instead of)
11/5/2008	5:02 PM	5:26 PM	5	19	Test	used isnumeric instead of t.eql?(numeric)
11/5/2008	5:30 PM	5:43 PM		13	Test	changed the isnumeric logic
11/7/2008	7:16 PM	7:43 PM		27	Code	Coded the calc_rightequation and calc_leftequation methods
11/7/2008	7:45 PM	7:53 PM		8	Test	Dynamic constant assignment

Table 16: Example Timelog

4.11.2 Changelog:

The changelog is another important artifact that is used to hold the data on the defects the developer find and correct. A defect is counted every time the developer change a program to fix a problem. The change could be one character or multiple statements.

The change log will record the following basic data:

1.) The defect type.

2.) The phase in which the defect was injected.

3.) The phase in which the defect was removed.

4.) The fix time.

5.) A brief description of the defect.

PCSE uses the PSP defect type standard [21]. Although this standard is quite simple, it should be sufficient to cover most of the developer's needs.

Documentation	**Comments, Message**
Syntax	spelling, punctuation, typos, improper programming language grammar
Build, package	change management, library, version control
Assignment	declaration, duplicate names, scope, limits
Interface	module calls and references, user formats
Checking	error messages, inadequate bound/type/format checking
Data	structure, content
Function	logic, pointers, loops, recursion, computation, functional defects
System	configuration, timing, memory
Environment	programming tools

Table 17: Defect Standard Type

The following are the important Changelog data and their instructions:

General: Record each defect separately and completely.

Type: Enter the defect type from the defect type list. Use the best judgment in selecting which type applies.

Inject: Enter the phase this defect was injected. Use the best judgment.

Remove: Enter the phase during which the developer fixes the defect. This will generally be the phase when the developer found the defect.

Inject Feature set: Enter the iteration this defect was injected.

Remove Feature set: Enter the iteration this defect was removed.

Fix Time: Enter the time the developer took to find and fix the defect. This time can be determined by stopwatch or judgment.

Fix Reference: If the developer injected this defect while fixing another defect, record the number of the improperly fixed defect. If the developer cannot find the defect number, enter an X.

Description: Write a succinct description of the defect that is clear enough to later remind the developer about the error and help the developer to remember why he made it.

The changelog is useful for the following:

1.) Deciding where and how to improve the developer's process.

2.) Observing fix time data can help the developer evaluate the cost and schedule consequences of his personal process improvements.

3.) Analyzing defects for pattern.

4.) Preventing defects.

5.) Creating a review checklist for commonly occurring defects.

The following is an example Changelog:

No.	Date	Type	Inject	Remove	Fix Time	Fix Ref.	Description
1	11/5/2008	Syntax	Code	Test	5		syntax error, unexpected $end, expecting kEND
2	11/5/2008	Function	Code	Test	9		used abs(n) instead of n.abs
3	11/5/2008	Syntax	Code	Test	4		Syntax error - usage of)) instead of)
4	11/5/2008	Function	Code	Test	19		used isnumeric instead of t.eql?(numeric)
5	11/5/2008	Checking	Test	Test	13	4	changed the isnumeric logic
6	11/7/2008	Assignment	Code	Test	8		Dynamic constant assignment
7	11/7/2008	Assignment	Test	Test	2	6	Dynamic constant assignment N = N*2
8	11/7/2008	Function	Code	Test	8		Used line.strip.split(//) instead of line.strip.split
9	11/7/2008	Function	Code	Test	43		Got error whilecomparing 2 float numbers. Had to use BigDecimal inorder to fix this issue
10	11/7/2008	Checking	Code	Test	31		Infinite while loop. Failed to multiply the value with W/3
11	11/7/2008	Checking	Code	Test	12		Made an integer divide
12	11/9/2008	Function	Test	Test	20		logic error - had to fix the repeateadly calling calc_gama function from calc_leftequation
13	11/9/2008	Data	Code	Test	8		Used Math.PI instead of Math::PI
14	11/9/2008	Syntax	Code	Test	2		Syntax error, unexpected ','
15	11/9/2008	Function	Test	Test	5		Error : in 'calc_gamma': undefined method '*' for nil: NilClass (NoMethodError)

Table 18: Example Changelog

PCSE Eclipse Plug-ins

5.1 Scope:

Although PCSE involves many activities and artifacts, the scope of the plug-in development was restricted to the timelog, changelog and the size estimation.

5.2 Development Environment:

The PCSE plug-ins (Timelog, Changelog, Size Estimation) that were developed were intended to be used by potential users who build/develop Java projects using the Eclipse IDE (Integrated Development Environment). The plug-ins were developed in Eclipse Standard Development Environment (Eclipse SDK version 3.5.2) using the Java 2 Standard Edition 6.0 (J2SE 6.0), Java 2 Enterprise Edition (J2EE) Application Programming Interfaces (APIs), Eclipse Platform API Specification and Standard Widget Toolkit (SWT), a graphical widget toolkit for use with the Java platform.

The Size Estimation plug-in connects to Auburn University’s master Apache Subversion (SVN) version control system, using Subclipse 1.6.13, an Eclipse Team Provider plug-in providing support for Subversion.

JavaHL 1.6.12 (JNI) (or) SVNKit 1.3.3 toolkits were also used. These are pure Java toolkits that implement all Subversion features. They provide APIs to work with Subversion working copies, access, and manipulate Subversion repositories.

5.3 PCSE Eclipse Plug-ins:

All the PCSE plug-ins were developed as Eclipse views and added to the workbench. In the Eclipse Platform, a view is typically used to navigate a hierarchy of information, open an editor, or display properties for the active editor.

The following were the major four PCSE Eclipse plug-in views developed and integrated to the Eclipse workbench:

1.) Process Dashboard

2.) Timelog

3.) Changelog

4.) Size Matrix

5.3.1 Process Dashboard:

The process dashboard view is one single layout where all controls (i.e. buttons, text boxes, etc.), graphs, information are displayed. It acts as a central view, where user can input data by clicking the interactive buttons present in it. The process dashboard interacts with the other views like timelog and size matrix, based on the input received from the user. The time log records values based on the input provided in the process dashboard. It contains the following controls, which are created using Java SWT toolkit.

Activities: represents the different activities of the PCSE life cycle. Every activity is created as a toggle button.

Comments: is a textbox to capture user comments

Feature Set: is created as a spinner, to represent the iteration.

Timer: is a toggle button that toggles between START and STOP.

Code Complete: is a button to trigger size matrix calculation

Flow Graph: is a graphical flow chart to indicate the current phase/activity. The background color changes according to the current activity selection.

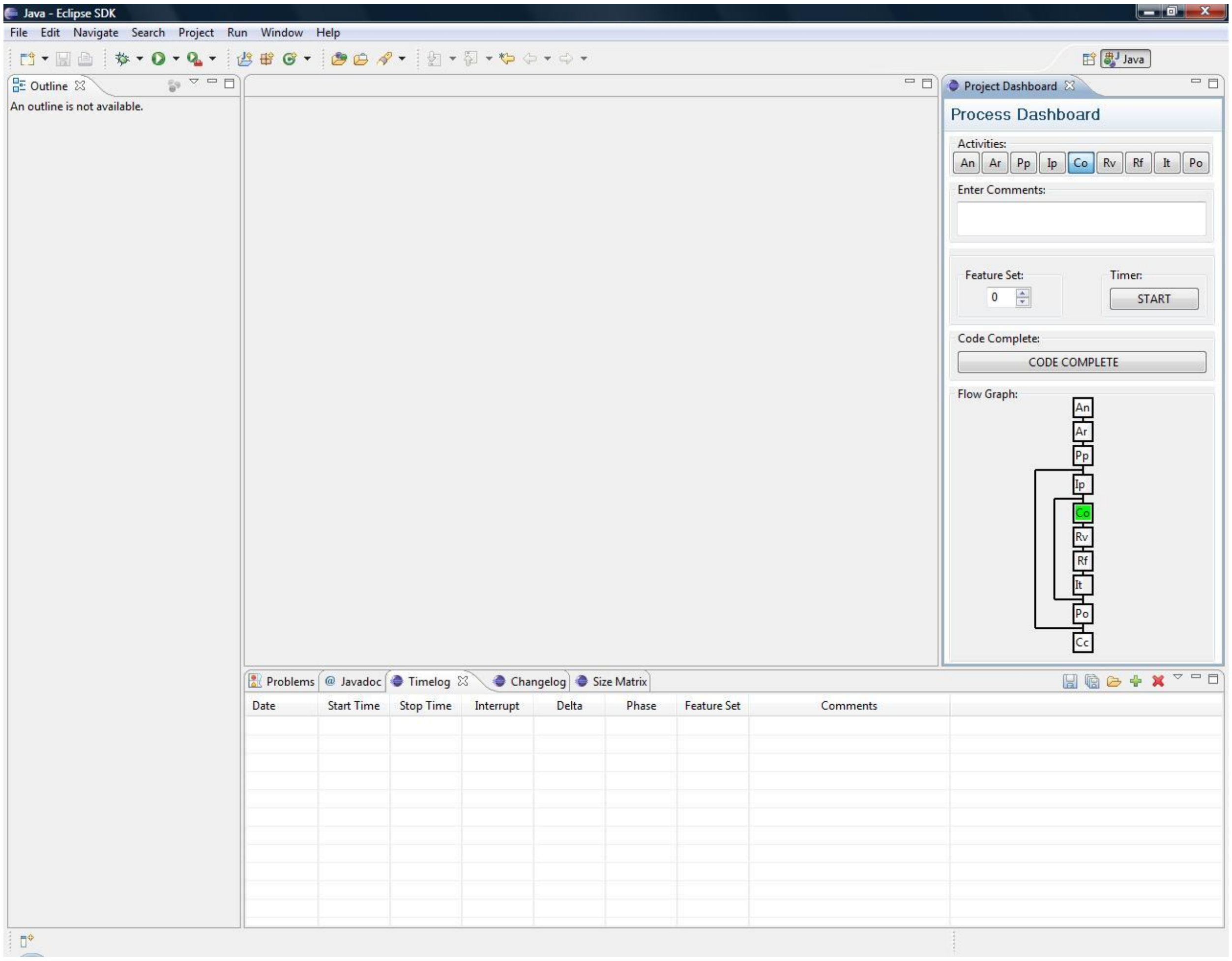


Figure 11: PCSE Process Dashboard

5.3.1.1 Property file:

All the configurable parameters including the different activities, the defect types are read from a .properties file. Each parameter is stored as a pair of strings, one storing the name of the parameter (called the key), and the other storing the value. This flexibility will allow to dynamically changing the activity or a defect type in the future. Also the order can be rearranged.

5.3.1.2 Using the Process Dashboard:

a.) Click on any activity, enter comments if necessary, and start the timer.

As soon as the timer is started the timer button toggles to STOP from START, to indicate the user that it can be stopped at any time. A new record is created in the timelog view with the following details: date, start time, phase and comments. The start time and the stop time are fetched from the system time.

b.) If the user wishes to switch to another activity/phase, he can click on the corresponding phase without stopping the timer, i.e., whenever users switch the activity, the stop time for the previous record is automatically recorded in the timelog and a new entry is created with the new activity. This flexibility allows the user to not stop and start the timer each and every time he moves on to a different activity.

c.) If the timer triggers at midnight, then a new record is automatically created with the next day's date in the Date field.

d.) The code complete button is used to trigger the size estimate calculation, i.e., each time the code complete button is pressed, the size estimate is recalculated and the size matrix view is populated with the latest values.

5.3.1.3 Passing data between views:

One of the greatest challenges of developing this plug-ins was to pass data between the views. One of the integration aspects are view parts that provide additional information for particular objects and update their content automatically whenever such objects are selected somewhere in the workbench window.

This was achieved by using “selection service”. It decouples parts where items can be selected from others reacting on selection changes.

Each workbench window has its own *selection service* instance. The service keeps track of the selection in the currently active part and propagates selection changes to all registered listeners. Such selection events occur when the selection in the current part is changed or when a different part is activated. Both can be triggered by user interaction or programmatically.

The view where elements or text are selected does not need to know who is interested in the selection; Therefore, new views can be created based on the selection of an existing view without changing a single line of code in that view [24].

This was achieved by implementing the selectionChanged() method of the ISelectionListener interface.

The following is a graphical representation of a Selection Service:

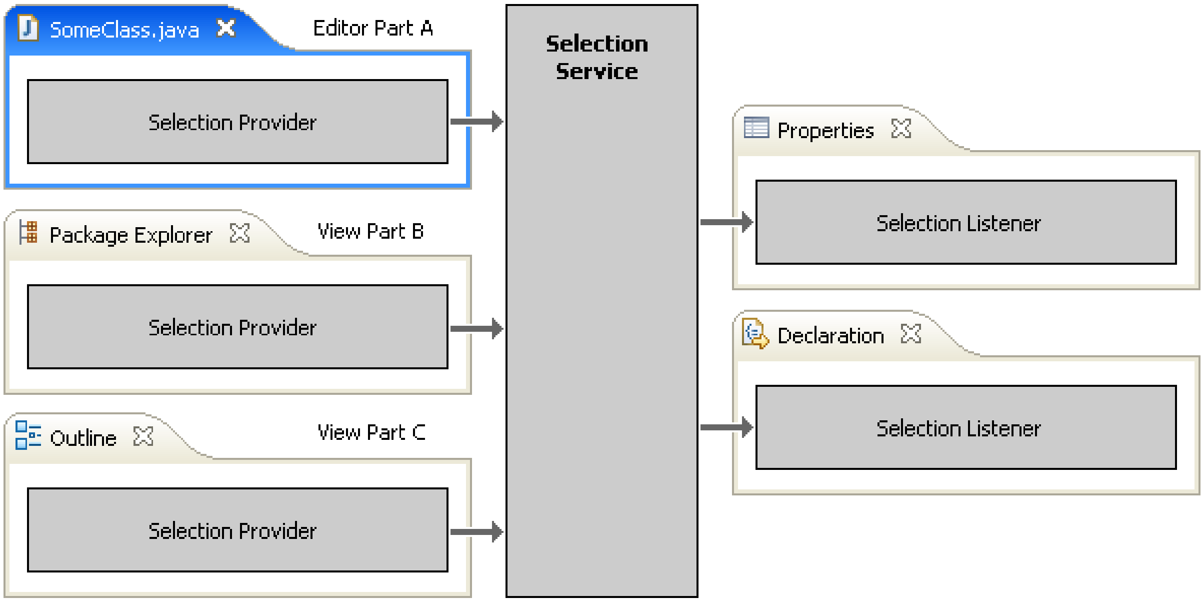


Figure 12: Selection service - The big picture

5.3.2 Timelog:

The timelog view records data based on the user interaction on the process dashboard. It was built using the Java SWT widget TableViewer, a concrete viewer based on a SWT Table control. Each column in a TableViewer can be treated as a control. The timelog TableViewer contains the following columns:

Warning Icon (!): Indicates a custom entry. Often there are times where users wish to add additional information. All such records are indicated using this warning icon. Currently this feature is not implemented in this version.

Date: Indicates the start date of an activity.

Start Time: Indicates the start time of an activity. It is read from the system time.

Stop Time: Indicates the stop time of an activity. It is read from the system time.

Interrupt: Record any interruption time (e.g. phone calls, questions or other things) that was spent on the process activity. Currently this feature is not implemented in this version.

Delta: it is the actual time spent working on the process activity, less the interruption time.

Phase: represents the activities. It is provided as a combo box and are read from the property file.

Feature Set: Represents the iteration. Currently this feature is not implemented in this version.

Comments: a field to capture any additional information.

5.3.2.1 Features:

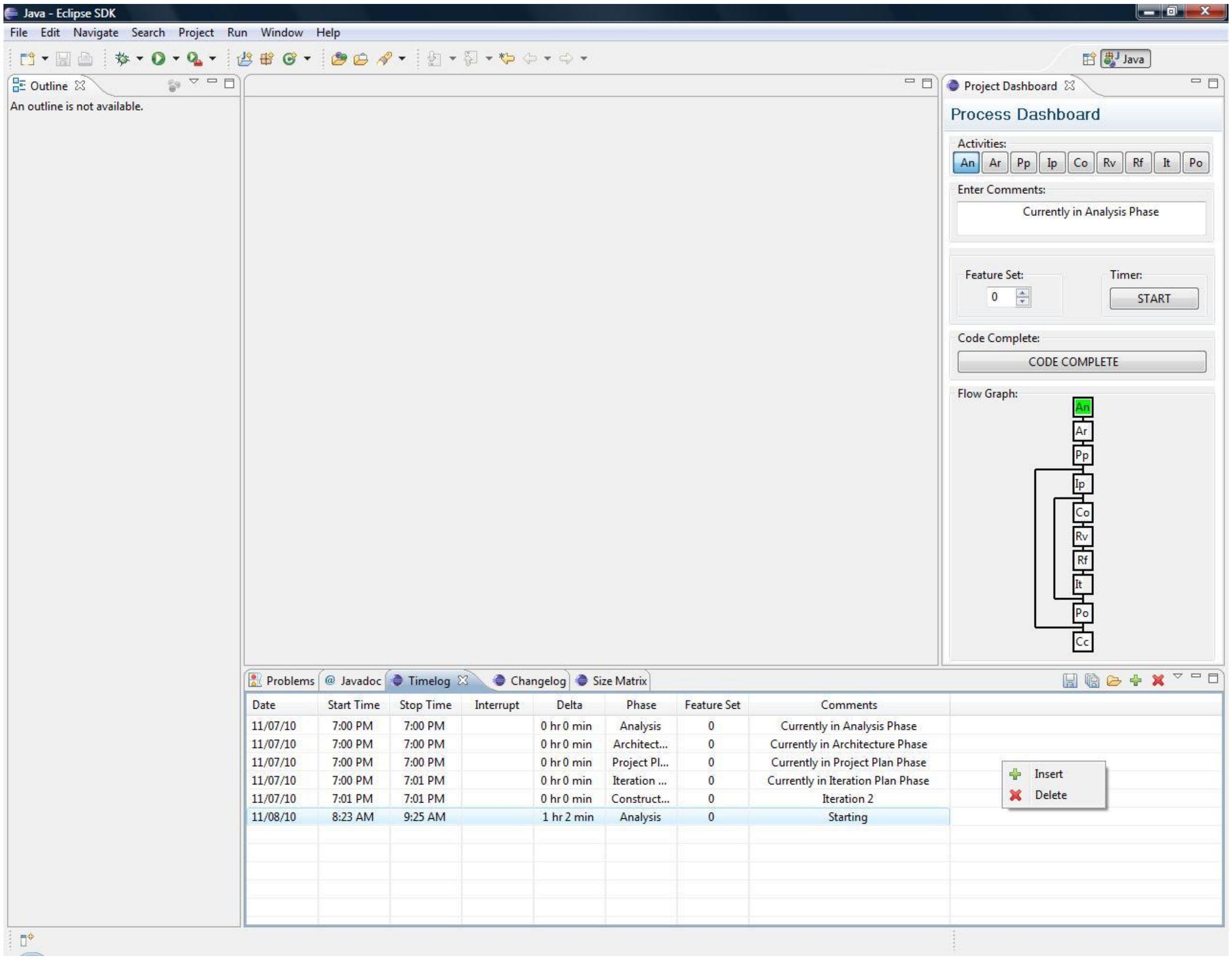
a.) Insert and Delete:

The timelog view has a provision to insert and delete new rows. For inserting, the user has to select a row to indicate that the new row should be created above the selected row. Similarly for deleting, the selected row will be deleted. All actions were achieved by using org.eclipse.jface.action.Action, a standard abstract implementation of an action.

b.) Save, SaveAs and Open:

The timelog allows the user to save and retrieve data at any given point. All data are stored and retrieved as a XML file. This was achieved by using the class JAXBContext().

The JAXBContext class provides the client's entry point to the JAXB API. It provides an abstraction for managing the XML/Java binding information necessary to implement the JAXB binding framework operations: unmarshal, marshal and validate.



An outline is not available.

Currently in Analysis Phase

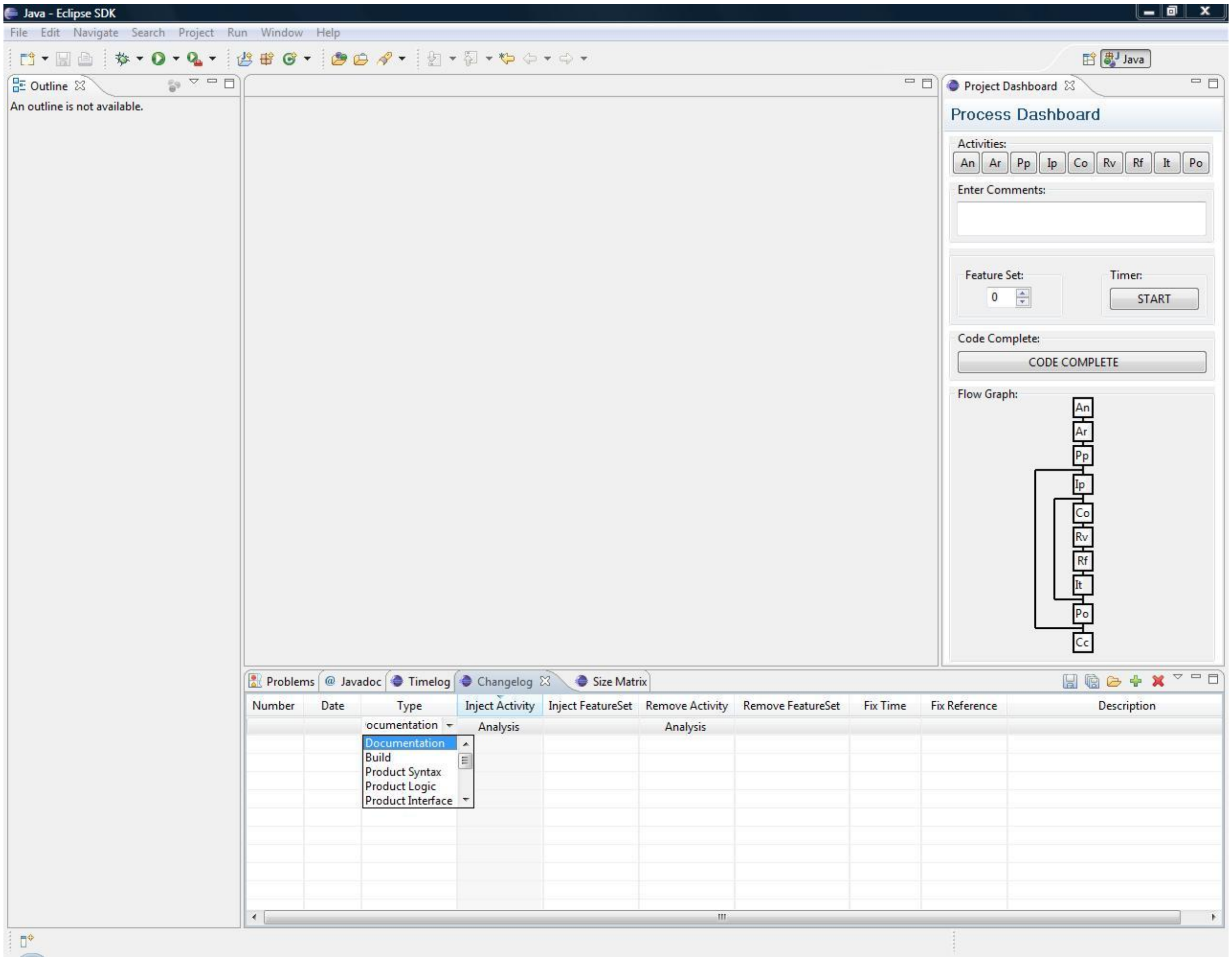
Feature Set: 0

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase
11/07/10	7:00 PM	7:01 PM		0 hr 0 min	Iteration ...	0	Currently in Iteration Plan Phase
11/07/10	7:01 PM	7:01 PM		0 hr 0 min	Construct...	0	Iteration 2
11/08/10	8:23 AM	9:25 AM		1 hr 2 min	Analysis	0	Starting

Figure 13: PCSE Timelog

5.3.3 Changelog:

The PCSE changelog is similar to the timelog as it was built using the Java SWT widget TableViewer. It has no interaction with the process dashboard and all entries are added manually by the user. It has support for Insert, Delete, Save, SaveAs and Open. The changelog TableViewer has the following columns: Number, Date, Type, Inject Activity, Inject FeatureSet, Remove Activity, Remove FeatureSet, FixTime, Fix Reference, Description. The definitions remain the same as how it is defined in Chapter 3 (changelog). The type , inject activity and remove activity are read from the properties file. Inject FeatureSet and Remove FeatureSet are not implemented in this version.



Number	Date	Type	Inject Activity	Inject FeatureSet	Remove Activity	Remove FeatureSet	Fix Time	Fix Reference	Description
		ocumentation	Analysis		Analysis				

Figure 14: PCSE Changelog

5.3.4 Size Matrix:

The PCSE size matrix has 2 layouts: a.) parts information, b.) size matrix. Both the layouts were built using the Java SWT widget TableViewer. The scope of this plug-in view is restricted to calculate parts information and size matrix. The size matrix is triggered/recalculated whenever user clicks “Code Complete” button in the project dashboard view. Although “Code

Complete" is to mark the end of the development as per the PCSE life cycle, here it has been used to trigger the size matrix calculation. This is because of the fact that at the end of every successful iteration and start of the next iteration one has to recalculate his estimation and schedule.

5.3.4.1 Parts Information:

The size matrix plug-in connects to the SVN source control, to retrieve and display the following historical projects information:

Projects: represent the historical projects.

Proxy Name: represent object-oriented class (or) functional method.

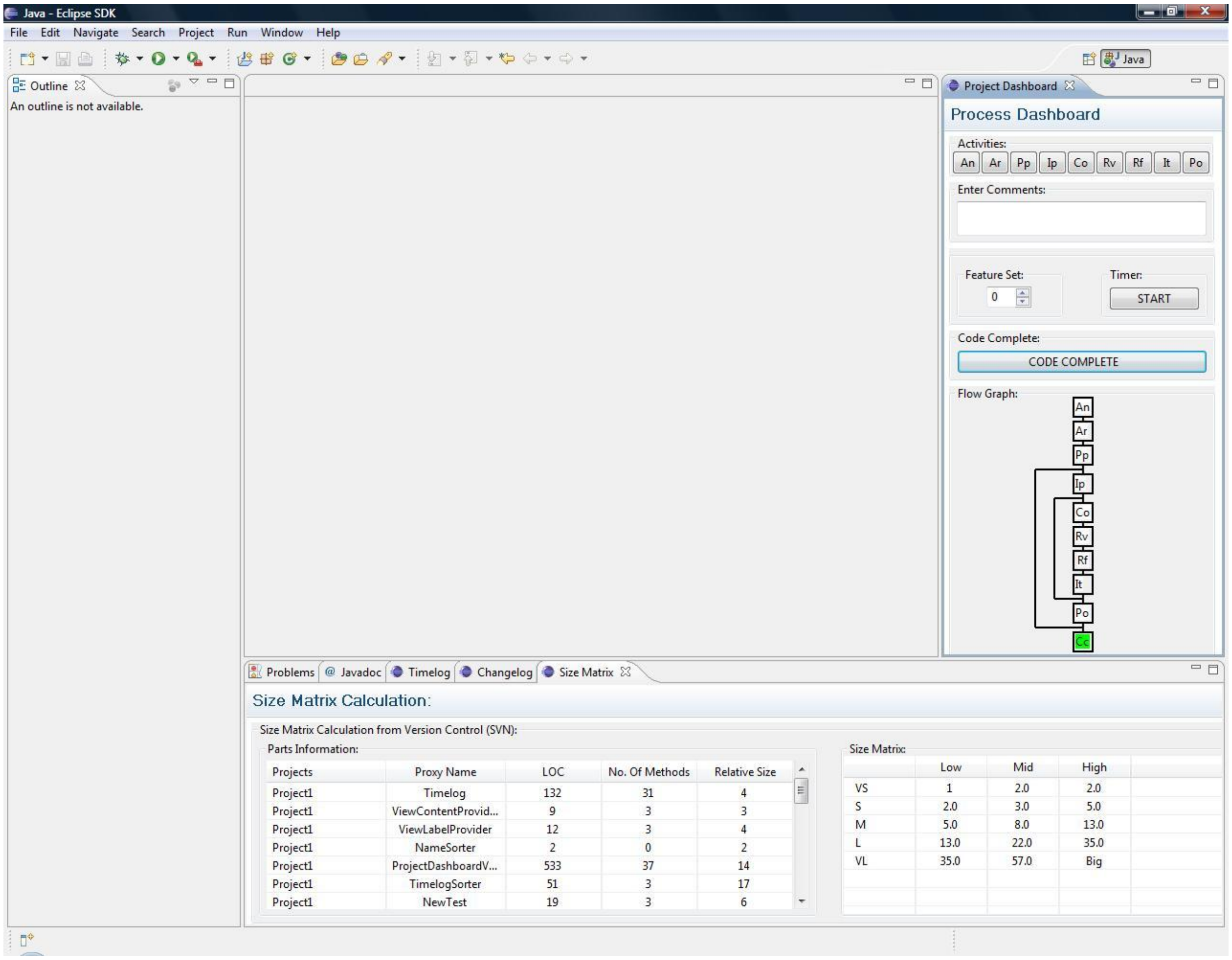
LOC: represents the Lines of Code per proxy.

No. Of Methods: represents the number of methods inside a proxy.

Relative Size: it is LOC/ No. Of Methods.

5.3.4.2 Size Matrix:

The size matrix is constructed based on the historical project data, which results in calculating the size ranges. The size ranges are categorized into Very Small (VS), Small (S), Medium (M), Large (L), and Very Large (VL). The size matrix will give the raw lines of code (LOCr) for the new project, from which the planned lines of code (LOCp) and the planned estimate time (Ep) can be calculated. Constructing the size matrix is discussed in details in chapter 3 (project plan).



An outline is not available.

Feature Set: 0

Size Matrix Calculation:

Size Matrix Calculation from Version Control (SVN):

Parts Information:

Projects	Proxy Name	LOC	No. Of Methods	Relative Size
Project1	Timelog	132	31	4
Project1	ViewContentProvid...	9	3	3
Project1	ViewLabelProvider	12	3	4
Project1	NameSorter	2	0	2
Project1	ProjectDashboardV...	533	37	14
Project1	TimelogSorter	51	3	17
Project1	NewTest	19	3	6

Size Matrix:

	Low	Mid	High
VS	1	2.0	2.0
S	2.0	3.0	5.0
M	5.0	8.0	13.0
L	13.0	22.0	35.0
VL	35.0	57.0	Big

Figure 15: PCSE Size Matrix

The following screen shot displays the historical projects from the SVN repository.

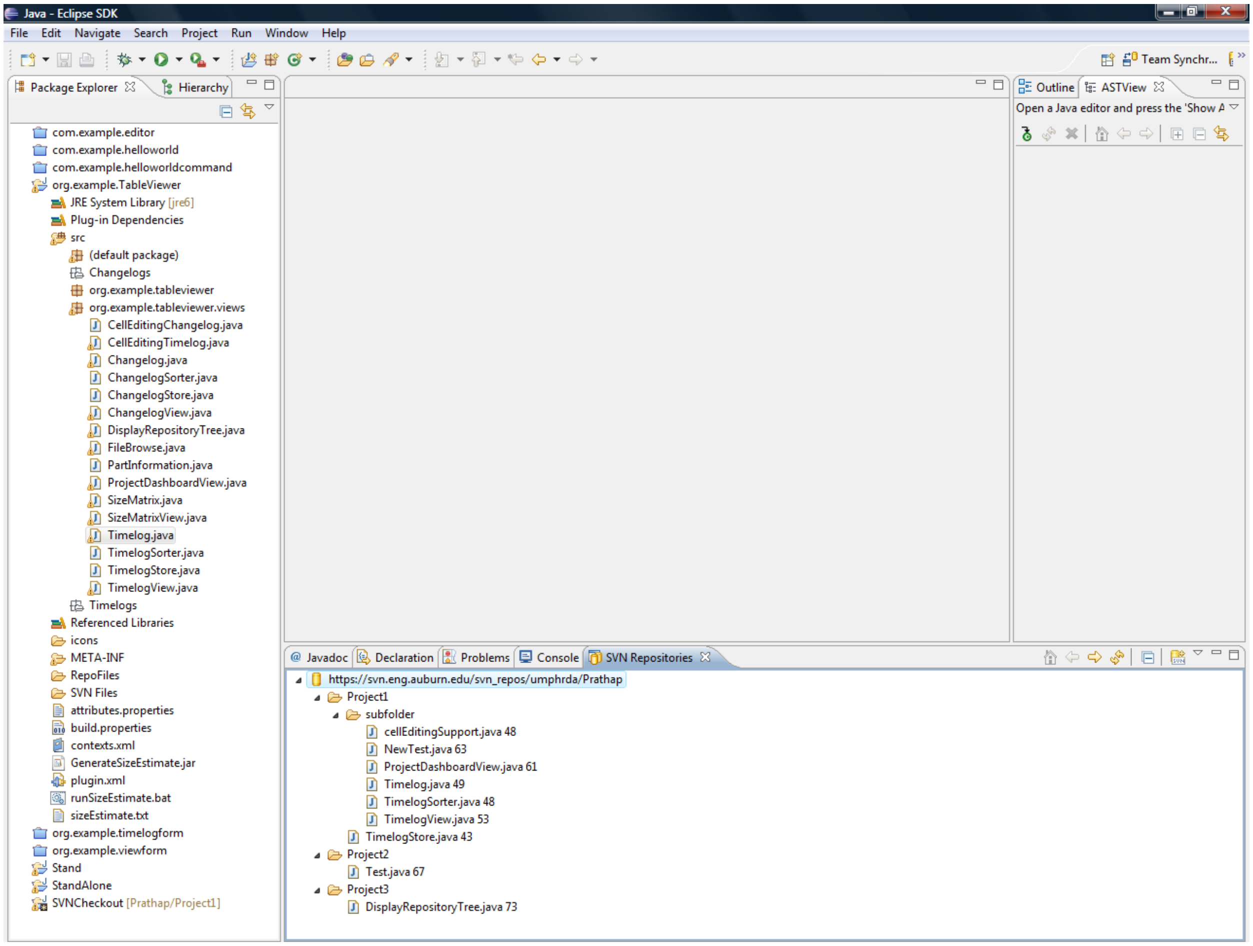


Figure 16: SVN Repository

5.3.4.3 Implementation:

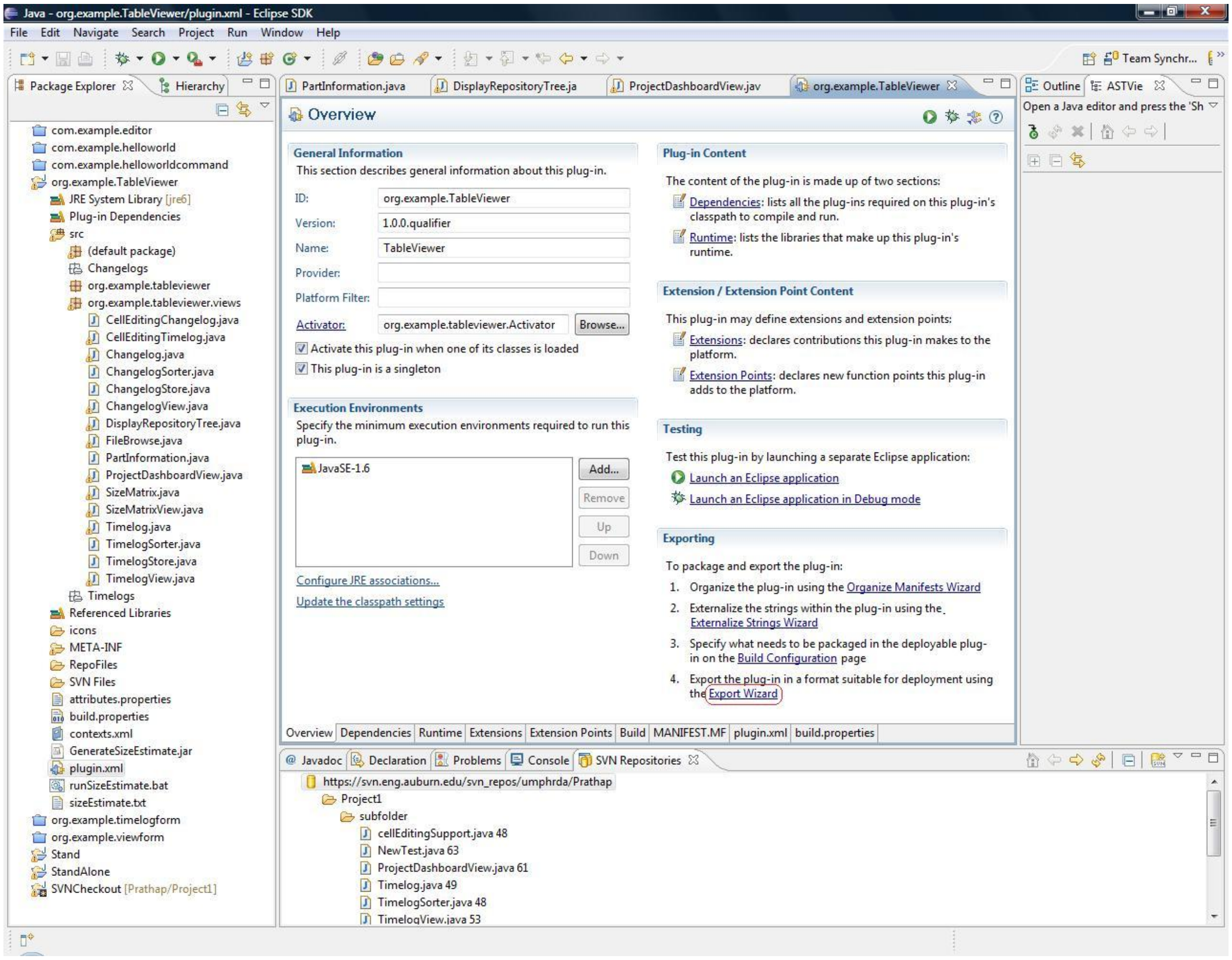
In order to retrieve and calculate the parts information and size matrix, the size matrix plug-in needs to connect to the SVN source control and read the information from the files. This can be achieved by using SVN interfaces from JavaHL 1.6.12 (JNI) or SVNKit 1.3.3. These are pure Java toolkits that implement all Subversion features and provide APIs to work with Subversion working copies, access and manipulate Subversion repositories [25].

Since all the methods in the above toolkits were exposed as factory abstract classes and methods, they did not work without modification. SVNKit was used to read java files from the repository. Custom parsing logic was written to get the parts information and calculate the size matrix.

The class DisplayRepositoryTree.java is a standalone class that gets triggered from within code using java.lang.Process (Runtime.exec() method), whenever the user clicks ‘Code Complete” button in the project dashboard view. The class connects to the SVN source control after reading the subversion login credentials from the .properties file. Upon successful connection it reads all the java files of the previous projects, parses each file, get the parts information and write it to a text file configured in the .properties file. The size matrix TableViewer reads the information from the text file and displays it in the size matrix view.

5.4 Deploying a Plug-in:

The plug-ins are deployed by putting them in an Eclipse drop-in folder [26]. This is done by selecting the plugin.xml file and activating the “Export Wizard”.



General Information

This section describes general information about this plug-in.

ID: org.example.TableViewer
Version: 1.0.0.qualifier
Name: TableViewer
Provider:
Platform Filter:
Activator: org.example.tableviewer.Activator

☑ Activate this plug-in when one of its classes is loaded
☑ This plug-in is a singleton

Execution Environments

Specify the minimum execution environments required to run this plug-in.

JavaSE-1.6

Configure JRE associations...
Update the classpath settings

Plug-in Content

The content of the plug-in is made up of two sections:

Dependencies: lists all the plug-ins required on this plug-in's classpath to compile and run.

Runtime: lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content

This plug-in may define extensions and extension points:

Extensions: declares contributions this plug-in makes to the platform.

Extension Points: declares new function points this plug-in adds to the platform.

Testing

Test this plug-in by launching a separate Eclipse application:

Launch an Eclipse application
Launch an Eclipse application in Debug mode

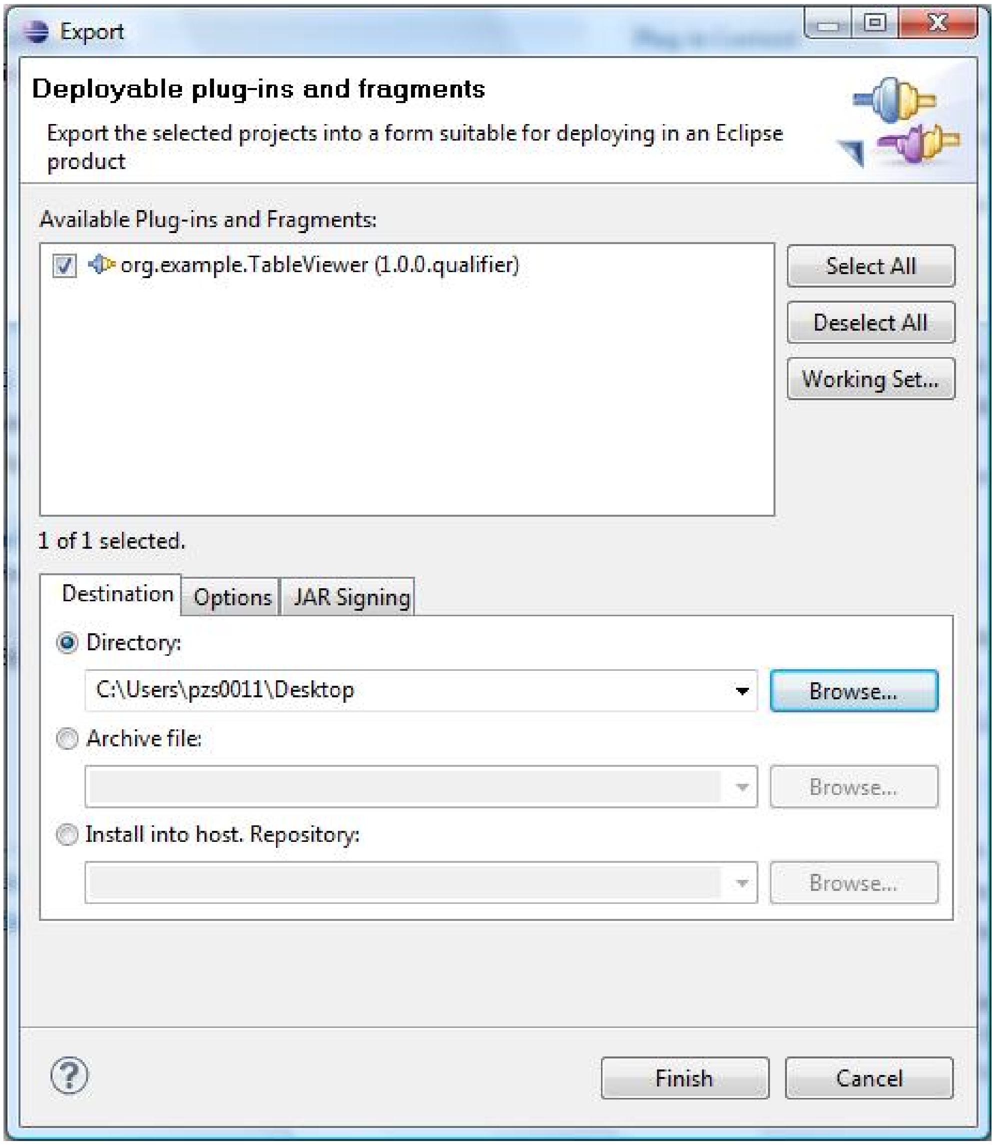
Exporting

To package and export the plug-in:

1. Organize the plug-in using the Organize Manifests Wizard
2. Externalize the strings within the plug-in using the Externalize Strings Wizard
3. Specify what needs to be packaged in the deployable plug-in on the Build Configuration page
4. Export the plug-in in a format suitable for deployment using the Export Wizard

https://svn.eng.auburn.edu/svn_repos/umphrda/Prathap
Project1
subfolder
cellEditingSupport.java 48
NewTest.java 63
ProjectDashboardView.java 61
Timelog.java 49
TimelogSorter.java 48
TimelogView.java 53

Figure 17: Plugin.xml’s overview tab



Export

Deployable plug-ins and fragments

Export the selected projects into a form suitable for deploying in an Eclipse product

Available Plug-ins and Fragments:

☑ org.example.TableViewer (1.0.0.qualifier)

1 of 1 selected.

Destination | Options | JAR Signing

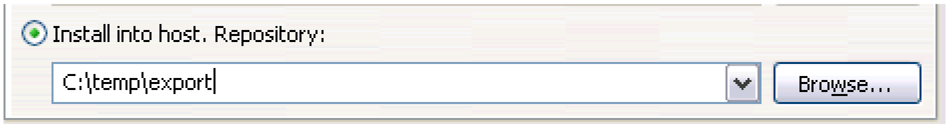
● Directory: C:\Users\pzs0011\Desktop

○ Archive file:

○ Install into host. Repository:

Figure 18: Plug-in deployment – export tab

This creates a jar in the directory plug-in. The jar is copied to the "dropin" directory in the Eclipse installation directory, Eclipse is restarted. During export one can also select "Install into host. Repository". This will export the new plugin into the selected directory (Repository) and install it directly into the running Eclipse.



● Install into host. Repository:

C:\temp\export

Figure 19: Install into host. Repository

One can also provide a plug-in via an update site [27].

Conclusion & Future Work

6.1 Conclusions:

This study was intended to develop a set of PCSE plug-ins that are integrated into the Eclipse environment. The plug-ins help an Eclipse user conveniently practice PCSE by reducing data gathering and analysis overhead. This study also provided a way to document the PCSE lifecycle and to explain the different activities of PCSE. The following are the major conclusions of this study:

1.) The successful deployment of the PCSE Eclipse plug-ins makes sure that it is possible to develop automated tools to support PCSE.

2.) The emergence of such plug-ins provided a way to the developers to largely reduce the overhead of manual metric collection.

3.) The PCSE size matrix plug-in reduces a lot of developer's time and effort, by calculating the size matrix in a single button click.

4.) The PCSE plug-ins important features like Save, SaveAs and Open allow the user to save and retrieve data. This data is helpful to measure and analyze the developer's productivity.

5.) The PCSE plug-ins were able to interact and fetch data from an external system, the Subversion source control in this case.

6.) The PCSE plug-ins give a confidence to the user by recording valid and accurate data.

6.2 Future Work:

Although the current PCSE plug-ins help an Eclipse user conveniently practice PCSE by reducing data gathering and analysis overhead, there is a room for incorporating more sophisticated features. The following are the important enhancements that can be incorporated in the current PCSE plug-ins:

1.) The current Timelog view does not have a support for calculating “Interrupt” and “Feature set”. Implementing these features will allow the user to analyze any interruption time (e.g. phone calls, questions or other things) that hindered Process activity.

2.) The current Changelog view does not have support for recording “Inject Feature set” and “Remove Feature set”. Implementing these features will allow the user to analyze work at the level of granularity of an iterative cycle.

3.) Whenever users click the “Code Complete” button in the current Process Dashboard, a batch file runs on a separate process to retrieve code from the Subversion source control, to count the number of lines of code, and to calculate the size matrix. The process terminates after writing the calculated size matrix into a text file. Once the process terminates, the Size Matrix view reads the data from the text file to populate the size matrix fields.

    This logic can be replaced by a more sophisticated Java API that works with Subversion to access and manipulate required parts information within Eclipse itself.

4.) The custom parsing logic to get the parts information from the SVN source control, to calculate the size matrix, can be replaced by a more sophisticated Java API or Java toolkit.

5.) The current Size Matrix view reads only .java files from the SVN source control. It can be redesigned to support files written in all languages.

With the development of new PCSE plug-ins to support other PCSE artifacts, a complete PCSE suite can be made available for the user, such as a.) An Eclipse view to calculate planned duration (Ep) and planned lines of code (LOCp). b.) An Eclipse view to support “Calendar” and “Burn down chart” etc. The PCSE plug-ins can also be developed to support Visual Studio environment.

References

[1] Software Process course notes Fall 09, By Dr. David A. Umphress, Computer Science and Software Engineering Department, Auburn University.

[2] "Personal Software Process". (2010, March 22). In Wikipedia, The Free Encyclopedia. Retrieved 23:27, April 18, 2010, from http://en.wikipedia.org/w/index.php?title=Personal_Software_Process&oldid=351366055

[3] "Heavyweight vs. lightweight methodologies" (2002, December 03), In Builder.com, By Jason P. Charvat, from http://www.builderau.com.au/strategy/projectmanagement/soa/Heavyweight-vs-lightweight-methodologies/0,339028292,320270383,00.htm

[4] Lightweight methodology. (2009, February 17). In Wikipedia, The Free Encyclopedia. Retrieved 20:49, October 31, 2010, from http://en.wikipedia.org/w/index.php?title=Lightweight_methodology&oldid=271435838

[5] "PSP Studio", By Dr. Joel Henry, Page last updated on 14th April, 97, Copyright © 1997 East Tennessee State University - http://csciwww.etsu.edu/psp/dlpsps.html

[6] “The Software Process Dashboard Initiative”, By Tuma Solutions LLC - http://www.processdash.com/

[7] “Eclipse (software)”. (2010, September 25). In Wikipedia, The Free Encyclopedia. Retrieved 20:36, September 27, 2010,
from http://en.wikipedia.org/w/index.php?title=Eclipse_(software)&oldid=386870797

[8] "Where did Eclipse come from?". Eclipse Wiki. Retrieved 2008-03-16, - http://wiki.eclipse.org/FAQ_Where_did_Eclipse_come_from%3F

[9] A Personal Software Process Tool for Eclipse Environment” by Xiaohong Yuan, Percy Vega, Huiming Yu, Yaohang Li, Department of Computer Science, North Carolina A&T State University - http://abner.ncat.edu/yaohang/publications/EclipsePSP2.pdf

[10] Easy Tracker Lite -Personal Time Manager 3.3 - http://www.download3000.com/download_21250.html

[11] PSP Log Control, version 4.0.4.1 - http://www.ipd.uka.de/mitarbeiter/muellerm/PSP/pplog/PSPLog_Control.doc

[12] Psptool, by Andrew M. Worsley – http://www.ipd.uka.de/mitarbeiter/muellerm/PSP/#psptool

[13] What is Correlation Coefficient, By 1997-2005. Financial Forecast Center LLC - http://forecasts.org/cc.htm

[14] IBM Rational Unified Process. (2010, September 27). In Wikipedia, The Free Encyclopedia. Retrieved 19:05, September 29, 2010, from - http://en.wikipedia.org/w/index.php?title=IBM_Rational_Unified_Process&oldid=387299444

[15] Burn down chart. (2010, August 31). In Wikipedia, The Free Encyclopedia. Retrieved 22:42, September 29, 2010, from - http://en.wikipedia.org/w/index.php?title=Burn_down_chart&oldid=382081147

[16] Test-driven development. (2010, September 23). In Wikipedia, The Free Encyclopedia. Retrieved 17:27, September 30, 2010, from http://en.wikipedia.org/w/index.php?title=Test-driven_development&oldid=386490917

[17] Test Automation Code Review Guidelines, By Devin A. Rychetnik, Microsoft Inc. - http://msdn.microsoft.com/en-us/library/ff519670.aspx

[18] Test Reviews Vs. Code Reviews - Some Helpful Tips, By Roy Osherove - http://weblogs.asp.net/rosherove/archive/2007/03/13/test-reviews-vs-code-reviews-some-helpful-tips.aspx

[19] Code refactoring. (2010, September 26). In Wikipedia, The Free Encyclopedia. Retrieved 12:57, October 1, 2010,

from http://en.wikipedia.org/w/index.php?title=Code_refactoring&oldid=387164240

[20] Regression testing. (2010, October 1). In Wikipedia, The Free Encyclopedia. Retrieved 13:41, October 1, 2010,

from http://en.wikipedia.org/w/index.php?title=Regression_testing&oldid=388072862

[21] PSP – A Self-Improvement Process for Software Engineers, by Watts S. Humphrey

[22] Your First Plug-in, Developing the Eclipse "Hello World" plug-in , By Jim Amsden,

Updated September 6, 2002 for Eclipse release 2.0 by Andrew Irvine

Last revised January 28, 2003 - http://www.eclipse.org/articles/Article-Your%20First%20Plug-in/YourFirstPlugin.html

[23] Creating an Eclipse View, By Dave Springgay,

November 2, 2001 - http://www.eclipse.org/articles/viewArticle/ViewArticle2.html

[24] Eclipse Workbench: Using the Selection Service, By Marc R. Hoffmann, Mountainminds GmbH & Co. - http://www.eclipse.org/articles/Article-WorkbenchSelections/article.html

[25] What is SVNKit - http://svnkit.com/

[26] Deploying a Plugin, By Lars Vogel -

http://www.vogella.de/articles/EclipsePlugIn/article.html

[27] Deploying a Plugin via an Update site, By Lars Vogel -

http://www.vogella.de/articles/EclipseP2Update/article.html

Appendix 1 (Creating an Eclipse plug-in)

Creating an Eclipse plug-in (Hello world example):

The following is a “Hello world” example plug-in illustrating the steps involved in creating the plug-in. This will be helpful for the reader to have an understanding of how to create a plug-in in an Eclipse environment. The steps originated from the article “Your First Plug-in, Developing the Eclipse ‘Hello World’ plug-in” , By Jim Amsden [22].

1. Introduction to Eclipse Platform:

In the Eclipse Platform the workbench contains a collection of workbench windows. Each workbench window contains one or more pages, and each page contains a collection of editors and views. An editor is typically used to edit or browse a document or input object. Modifications made in an editor follow an open-save-close lifecycle model. A view is typically used to navigate a hierarchy of information, open an editor, or display properties for the active editor. In contrast to an editor, modifications made in a view are saved immediately. The layout of editors and views within a page is controlled by the active perspective.

The workbench contains a number of standard components which demonstrate the role of a view. For instance, the Navigator view is used to display and navigate through the workspace. If a developer selects a file in the Navigator, he can open an editor on the contents of the file. Once an editor is open, he can navigate the structure in the editor data using the Outline view, or edit the properties of the file contents using the Properties view [23].

2. Integrating with Eclipse:

All extensions to Eclipse are done through plug-ins, and plug-ins integrates with each other through extensions on extension points. Eclipse plug-ins typically provides extensions to the platform that support some additional capability or semantics. What is needed is a way for plug-ins to allow other plug-ins to change their behavior in a controlled manner.

Eclipse provides an extensibility mechanism that is scalable, avoids name collisions, doesn't require compilation of the whole product as a unit, and supports multiple versions of the same component at the same time. Eclipse does this by introducing the notion of a plug-in which encapsulates functional extensions to the Platform. Each plug-in has a name, id, provider name, version, a list of other required plug-ins, and a specification for its runtime. A plug-in can also have any number of extension points that provide a portal into which other plug-ins can add their functionality. This is how Eclipse enables other plug-ins to handle the variability supported by your plug-in. In order to integrate with other plug-ins, a plug-in provides extensions on these extension points (perhaps even its own extension points in order to provide some default behavior).

A plug-in is described in an XML file called the plug-in manifest file. This file is always called plugin.xml, and is always contained in the plug-in sub-directory. The Eclipse Platform reads these manifest files and uses the information to populate and/or update a registry of information that is used to configure the whole platform [22].

3. Adding a new view:

A new view is added to the workbench using a simple three step process [23].

a.) Create a plug-in.

b.) Add a view extension to the plugin.xml file.

c.) Define a view class for the extension within the plug-in.

To illustrate this process we'll define a view called "Label", which just displays the words "Hello World".

3.1 Create a plug-in:

Create a plug-in project, and then define a plugin.xml file (shown below). This file contains a declaration of the plug-in id, name, pre-requisites, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<plugin
  name="Views Plugin"
  id="org.eclipse.ui.articles.views"
  version="1.0.0"
  provider-name="OTI">
    <requires>
        <import plugin="org.eclipse.core.boot"/>
        <import plugin="org.eclipse.core.runtime"/>
        <import plugin="org.eclipse.core.resources"/>
        <import plugin="org.eclipse.swt"/>
        <import plugin="org.eclipse.ui"/>
    </requires>
    <runtime>
        <library name="views.jar"/>
    </runtime>
</plugin>
```

3.2 Add a view extension to the plugin.xml file:

Once a plug-in is created, a view extension can be defined. The org.eclipse.ui plug-in defines a single extension point for view contribution: org.eclipse.ui.views. In the XML below, this extension point is used to add the Label view extension. This XML is added to the plugin.xml file, after the runtime element, and contains the basic attributes for the view: id, name, icon and class.

```
<extension point="org.eclipse.ui.views">
     <view id="org.eclipse.ui.articles.views.labelview"
          name="Label View"
A▶            class="org.eclipse.ui.articles.views.LabelView"
B▶            icon="icons\view.gif"/>
</extension>
```

A complete description of the view extension point and the syntax are available in the developer documentation for org.eclipse.ui. The attributes are described as follows.

- **id** - a unique name that will be used to identify this view
- **name** - a translatable name that will be used in the UI for this view
- **class** - a fully qualified name of the class that implements org.eclipse.ui.IViewPart. A common practice is to subclass org.eclipse.ui.part.ViewPart in order to inherit the default functionality.
- **icon** - a relative name of the icon that will be associated with the view.

Perhaps the most important attribute is class (A). This attribute must contain the fully qualified name of a class that implements org.eclipse.ui.IViewPart. The IViewPart interface defines the minimal responsibilities of the view, the chief one being to create an SWT Control within the workbench. This provides the presentation for an underlying model. In the XML above, the class for the Label view is defined as org.eclipse.ui.articles.views.LabelView. The implementation of LabelView will be examined in a few paragraphs.

The icon attribute (B) contains the name of an image that will be associated with the view. This image must exist within the plugin directory or one of the sub directories.

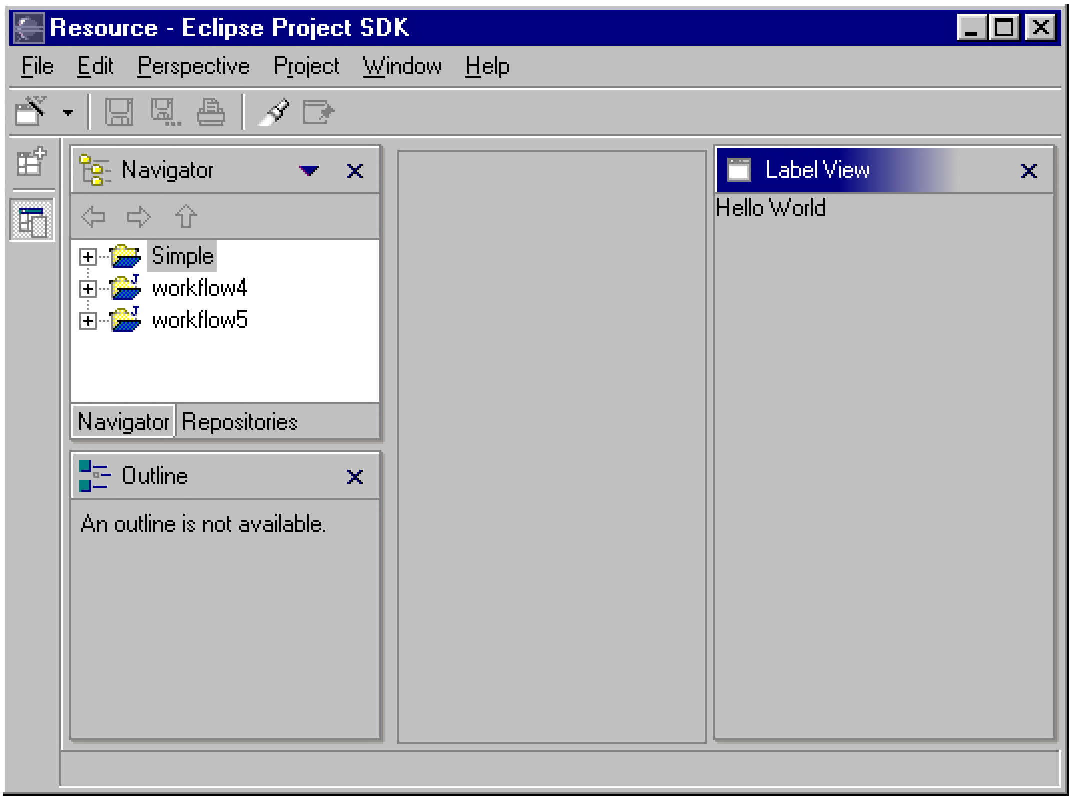
3.3 Define a view class for the Extension within the Plug-in:

Now the view class need to be defined. To define a view class, the platform contains an abstract class, named org.eclipse.ui.part.ViewPart, which implements most of the default behavior for an IViewPart. By subclassing this, one can inherit all of the behavior. The result is LabelView (shown below). The methods in this class implement the view specific presentation. The createPartControl method (B) creates an SWT Label object to display the phrase "Hello World". The setFocus (A) method gives focus to this control. Both of these methods will be called by the platform.

```
package org.eclipse.ui.articles.views;
import org.eclipse.swt.widgets.*;
import org.eclipse.ui.part.ViewPart;
public class LabelView extends ViewPart {
    private Label label;
    public LabelView() {
```

```
                super();
            }
A           public void setFocus() {
                label.setFocus();
            }
B           public void createPartControl(Composite parent) {
                label = new Label(parent, 0);
                label.setText("Hello World");
           }
        }
```

Once the LabelView class has been declared and compiled, the results can be tested. To do this, invoke the Perspective > Show View > Other menu item. A dialog will appear containing all of the view extensions: the Label View item will appear in the Other category. One should select the Label View item and press OK, the view will be instantiated and opened in the current perspective. Here is a screen snapshot of the Label view after it has been opened within the Resource perspective.



Hello World

An outline is not available.

Figure 20: Label view Hello World example plug-in

4. View Lifecycle:

The lifecycle of a view begins when the view is added to a workbench page. This can occur during the creation of the page, or afterwards, if the user invokes Perspective > Show View. In either case the following lifecycle is performed.

1.) On startup, the view id is mapped to a view extension within the plugin registry. Then a new instance of the view class is instantiated.. If the view class does not implement IViewPart an PartInitException is thrown.

2.) The IViewPart.init method is called to initialize the context for the view. An IViewSite object is passed, and contains methods to get the containing page, window, and other services is passed.

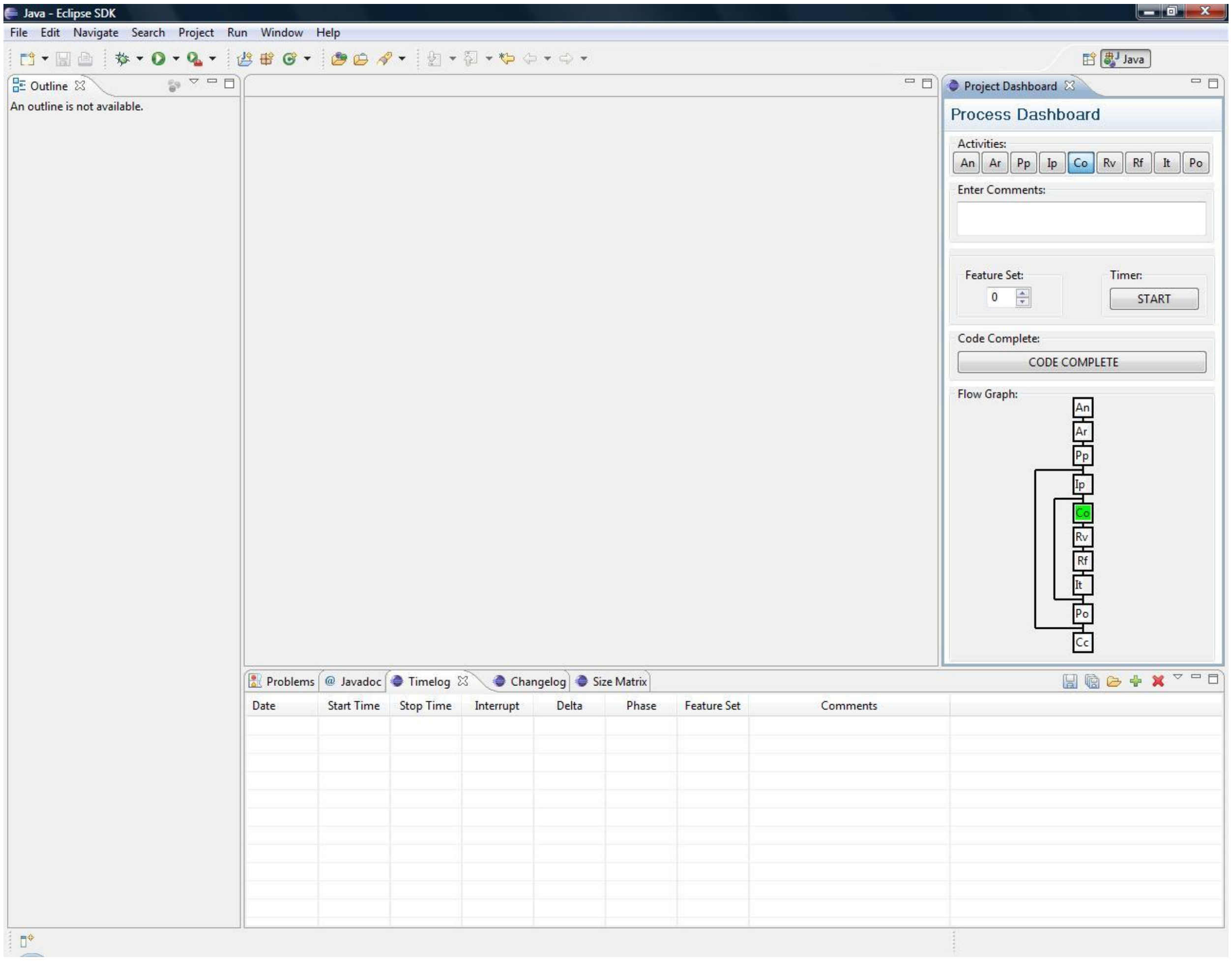
3.) The IViewPart.createPartControl method is called. Within this method the view can create any number of SWT controls within a parent Composite. These provide the presentation for some underlying, view specific model.

When the view is closed the lifecycle is completed.

1.) The parent Composite passed to createPartControl is disposed. This children are also implicitly disposed. If any code should be run at this time, it must hook the control dispose event.

2.) The IViewPart.dispose method is called to terminate the part lifecycle. This is the last method which the workbench will call on the part. It is an ideal time to release any fonts, images, etc.

Appendix 2 (Screen shots)

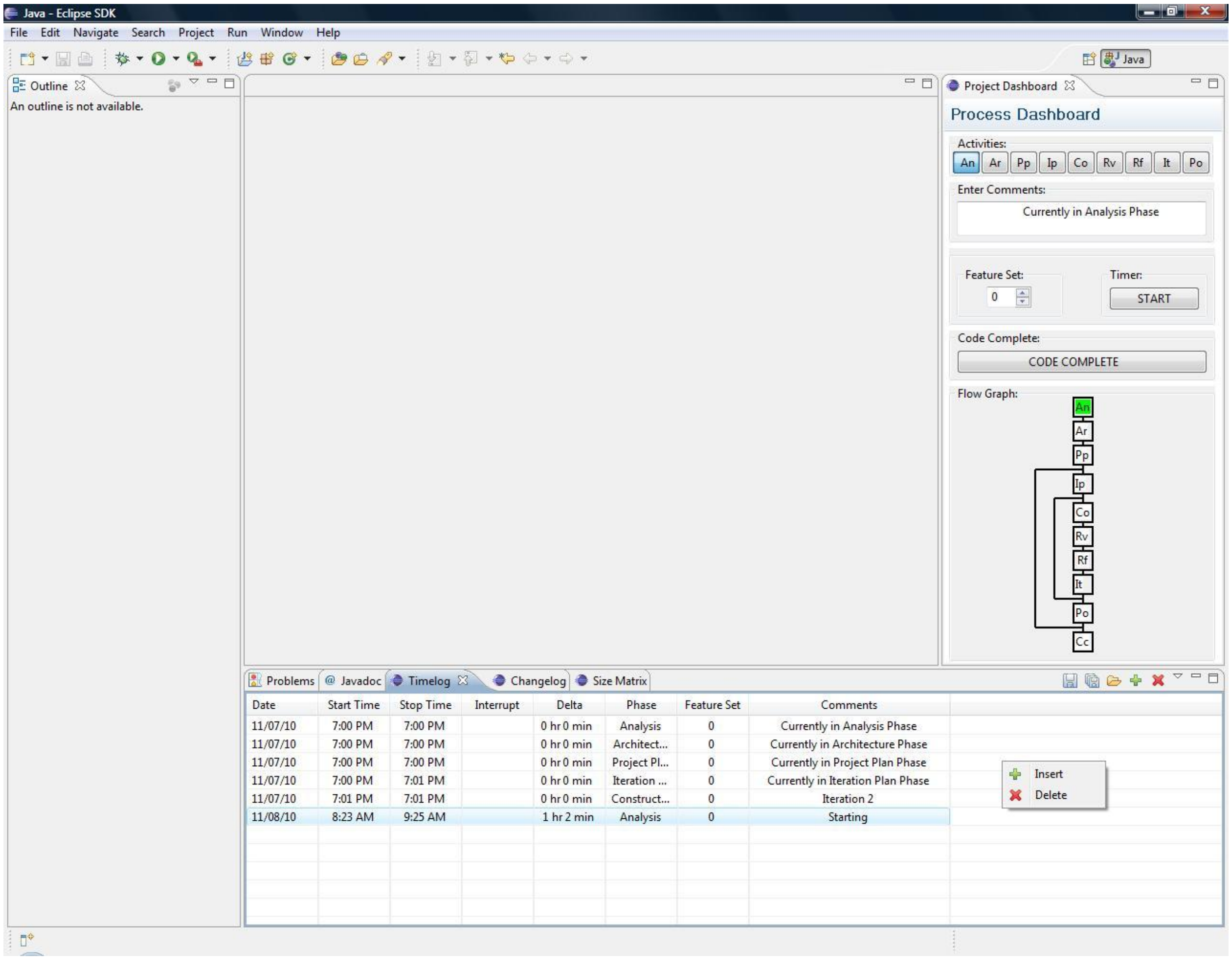


An outline is not available.

Feature Set: 0

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments

Screen shot 1: PCSE Process Dashboard



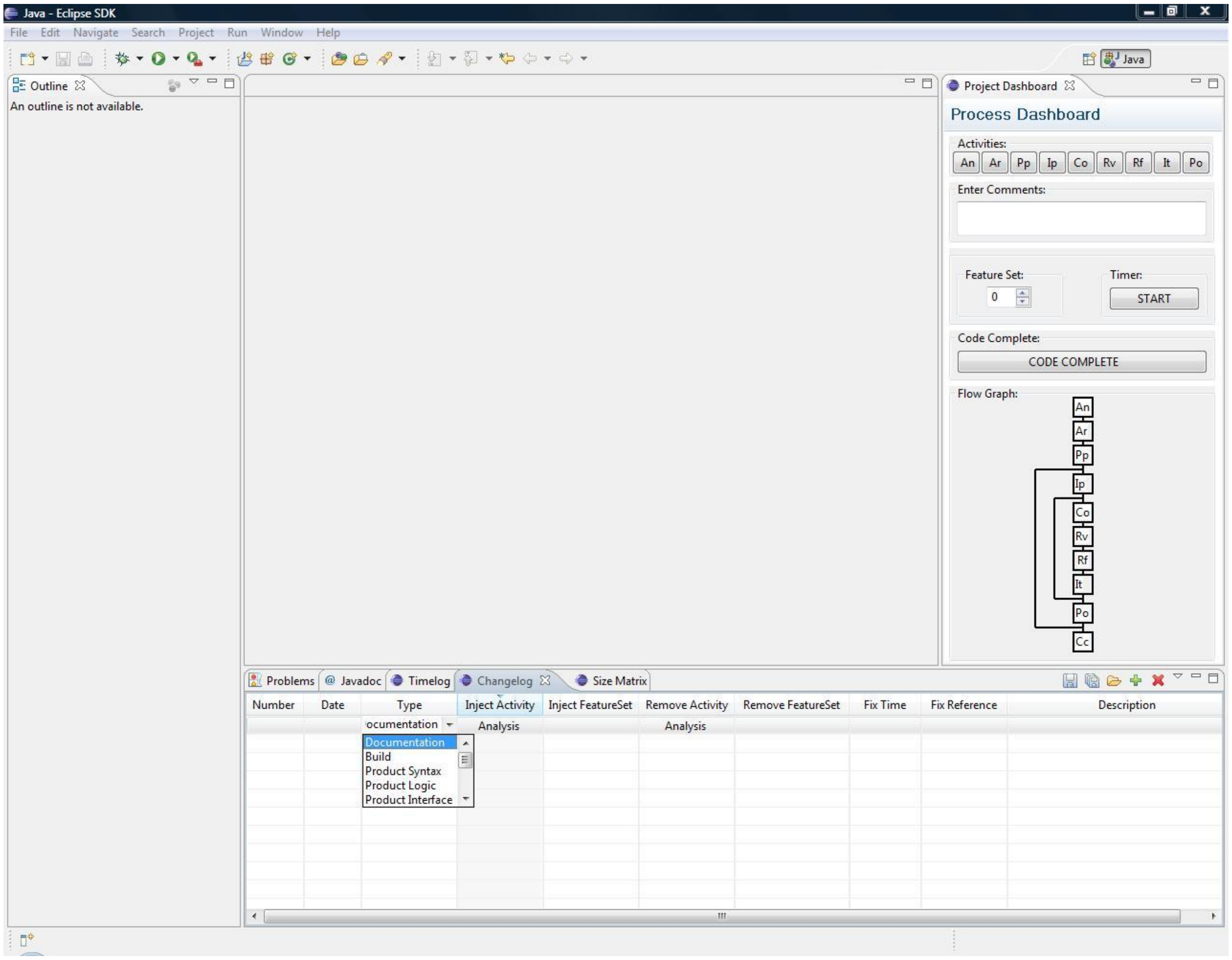
An outline is not available.

Enter Comments: Currently in Analysis Phase

Feature Set: 0

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase
11/07/10	7:00 PM	7:01 PM		0 hr 0 min	Iteration ...	0	Currently in Iteration Plan Phase
11/07/10	7:01 PM	7:01 PM		0 hr 0 min	Construct...	0	Iteration 2
11/08/10	8:23 AM	9:25 AM		1 hr 2 min	Analysis	0	Starting

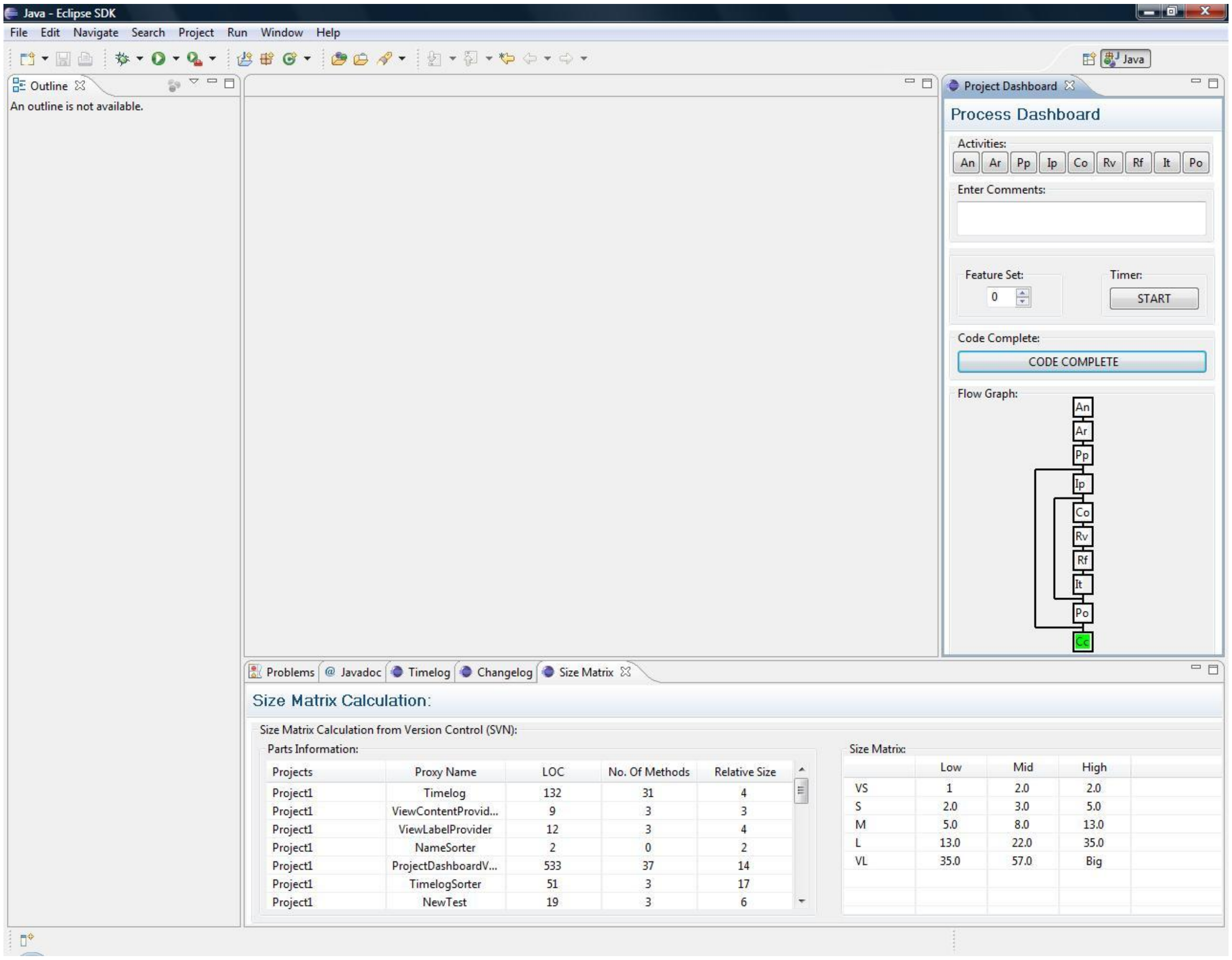
Screen shot 2: PCSE Timelog



Feature Set: 0

Number	Date	Type	Inject Activity	Inject FeatureSet	Remove Activity	Remove FeatureSet	Fix Time	Fix Reference	Description
		ocumentation	Analysis		Analysis				

Screen shot 3: PCSE Changelog



An outline is not available.

Feature Set: 0

Size Matrix Calculation:

Size Matrix Calculation from Version Control (SVN):

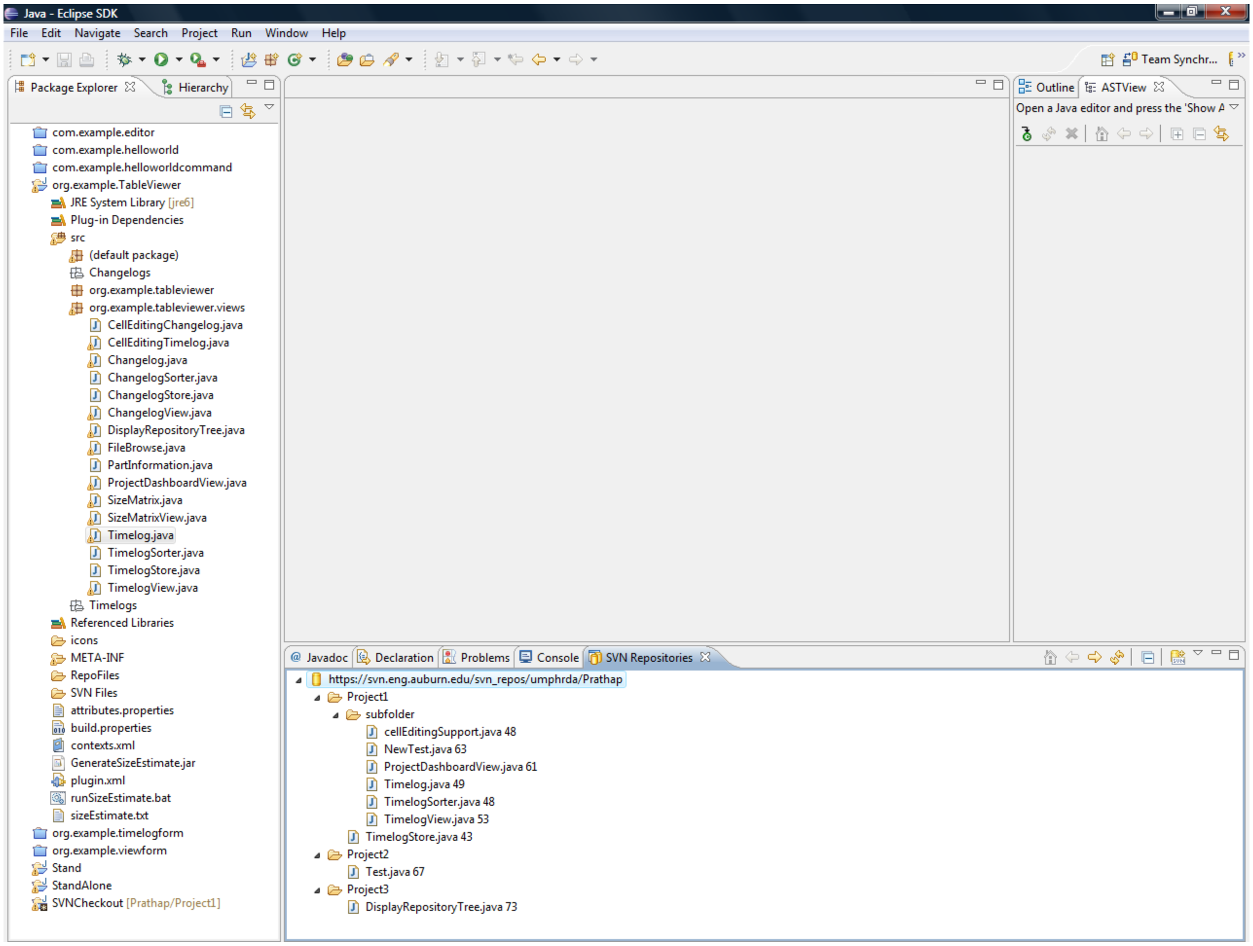
Parts Information:

Projects	Proxy Name	LOC	No. Of Methods	Relative Size
Project1	Timelog	132	31	4
Project1	ViewContentProvid...	9	3	3
Project1	ViewLabelProvider	12	3	4
Project1	NameSorter	2	0	2
Project1	ProjectDashboardV...	533	37	14
Project1	TimelogSorter	51	3	17
Project1	NewTest	19	3	6

Size Matrix:

	Low	Mid	High
VS	1	2.0	2.0
S	2.0	3.0	5.0
M	5.0	8.0	13.0
L	13.0	22.0	35.0
VL	35.0	57.0	Big

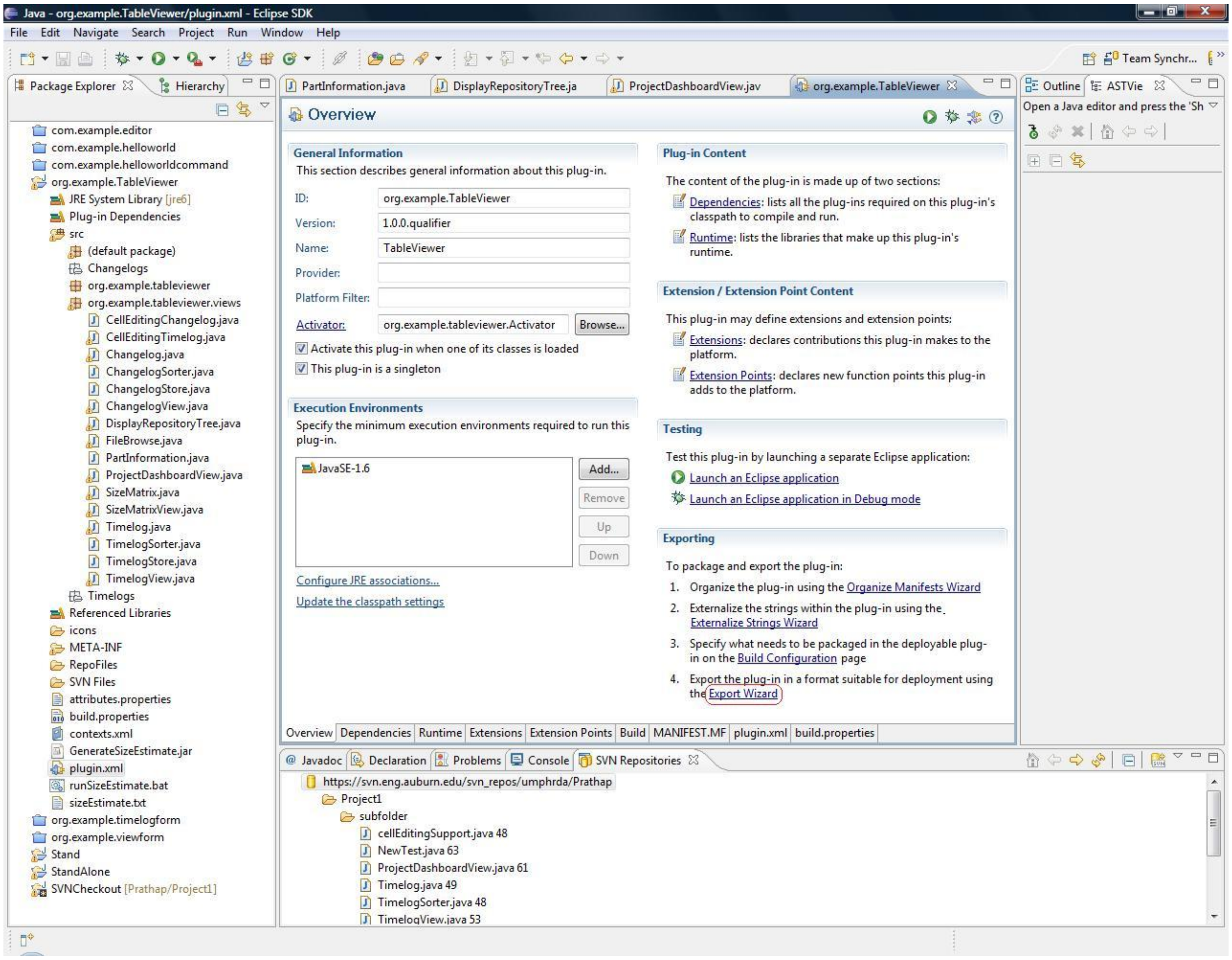
Screen shot 4: PCSE Size Matrix



```
com.example.editor
com.example.helloworld
com.example.helloworldcommand
org.example.TableViewer
    JRE System Library [jre6]
    Plug-in Dependencies
    src
        (default package)
        Changelogs
        org.example.tableviewer
        org.example.tableviewer.views
            CellEditingChangelog.java
            CellEditingTimelog.java
            Changelog.java
            ChangelogSorter.java
            ChangelogStore.java
            ChangelogView.java
            DisplayRepositoryTree.java
            FileBrowse.java
            PartInformation.java
            ProjectDashboardView.java
            SizeMatrix.java
            SizeMatrixView.java
            Timelog.java
            TimelogSorter.java
            TimelogStore.java
            TimelogView.java
        Timelogs
    Referenced Libraries
    icons
    META-INF
    RepoFiles
    SVN Files
    attributes.properties
    build.properties
    contexts.xml
    GenerateSizeEstimate.jar
    plugin.xml
    runSizeEstimate.bat
    sizeEstimate.txt
org.example.timelogform
org.example.viewform
Stand
StandAlone
SVNCheckout [Prathap/Project1]
```

```
https://svn.eng.auburn.edu/svn_repos/umphrda/Prathap
    Project1
        subfolder
            cellEditingSupport.java 48
            NewTest.java 63
            ProjectDashboardView.java 61
            Timelog.java 49
            TimelogSorter.java 48
            TimelogView.java 53
        TimelogStore.java 43
    Project2
        Test.java 67
    Project3
        DisplayRepositoryTree.java 73
```

Screen shot 5: SVN Repository



General Information

This section describes general information about this plug-in.

ID: org.example.TableViewer

Version: 1.0.0.qualifier

Name: TableViewer

Provider:

Platform Filter:

Activator: org.example.tableviewer.Activator Browse...

☑ Activate this plug-in when one of its classes is loaded

☑ This plug-in is a singleton

Execution Environments

Specify the minimum execution environments required to run this plug-in.

JavaSE-1.6

Add... Remove Up Down

Configure JRE associations...

Update the classpath settings

Plug-in Content

The content of the plug-in is made up of two sections:

Dependencies: lists all the plug-ins required on this plug-in's classpath to compile and run.

Runtime: lists the libraries that make up this plug-in's runtime.

Extension / Extension Point Content

This plug-in may define extensions and extension points:

Extensions: declares contributions this plug-in makes to the platform.

Extension Points: declares new function points this plug-in adds to the platform.

Testing

Test this plug-in by launching a separate Eclipse application:

Launch an Eclipse application

Launch an Eclipse application in Debug mode

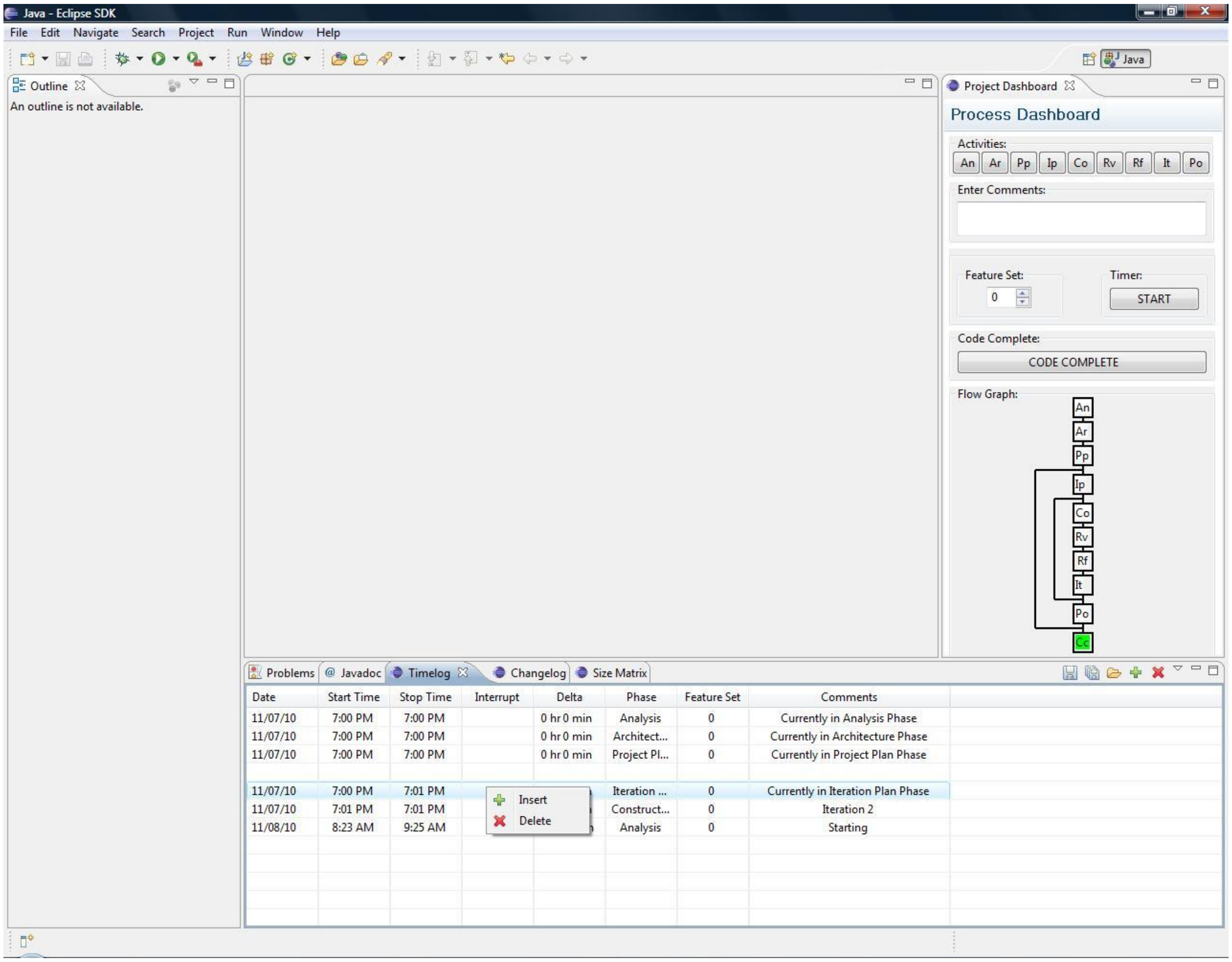
Exporting

To package and export the plug-in:

1. Organize the plug-in using the Organize Manifests Wizard
2. Externalize the strings within the plug-in using the Externalize Strings Wizard
3. Specify what needs to be packaged in the deployable plug-in on the Build Configuration page
4. Export the plug-in in a format suitable for deployment using the Export Wizard

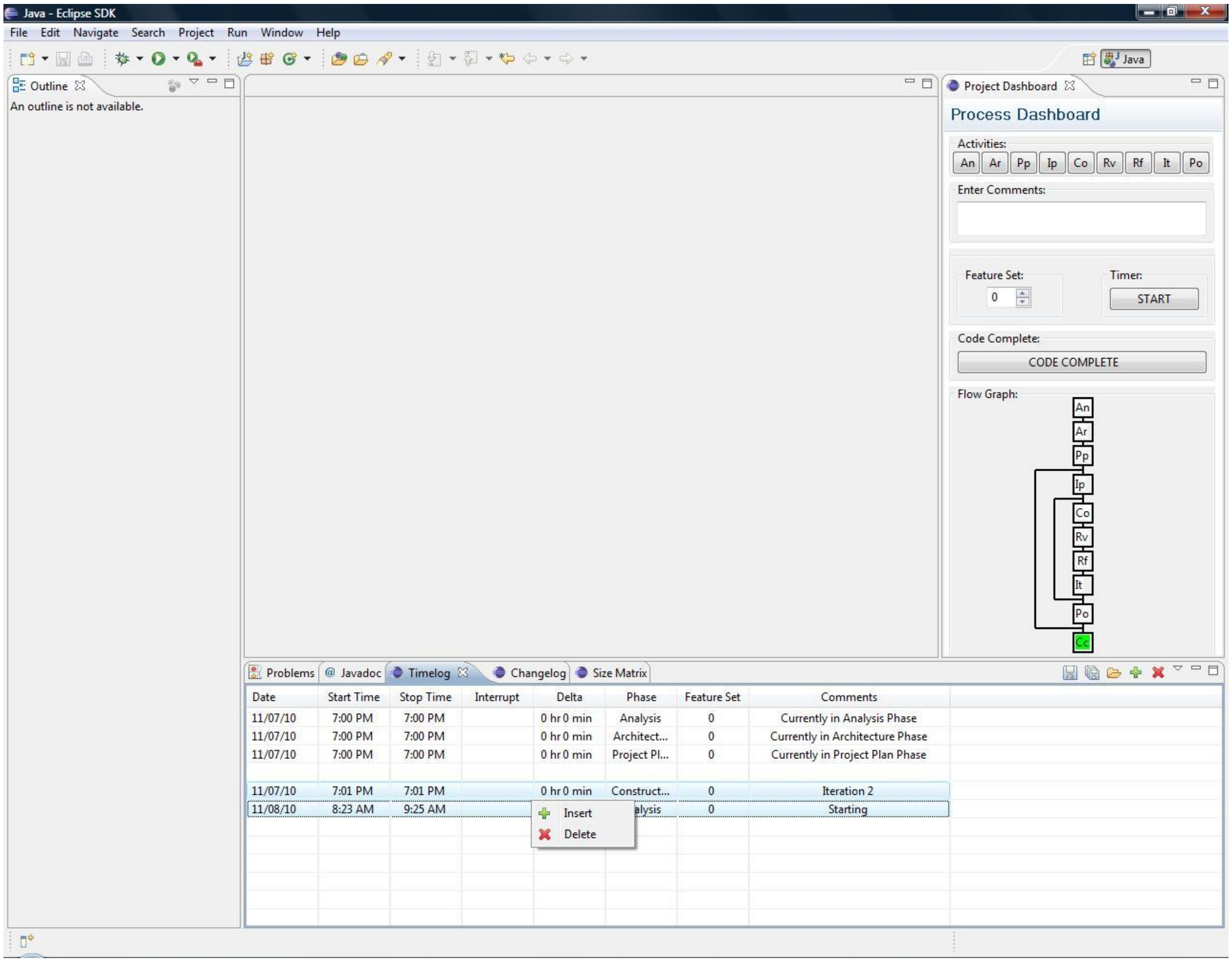
```
https://svn.eng.auburn.edu/svn_repos/umphrda/Prathap
  Project1
    subfolder
      cellEditingSupport.java 48
      NewTest.java 63
      ProjectDashboardView.java 61
      Timelog.java 49
      TimelogSorter.java 48
      TimelogView.java 53
```

Screen shot 6: The plugin.xml's overview tab



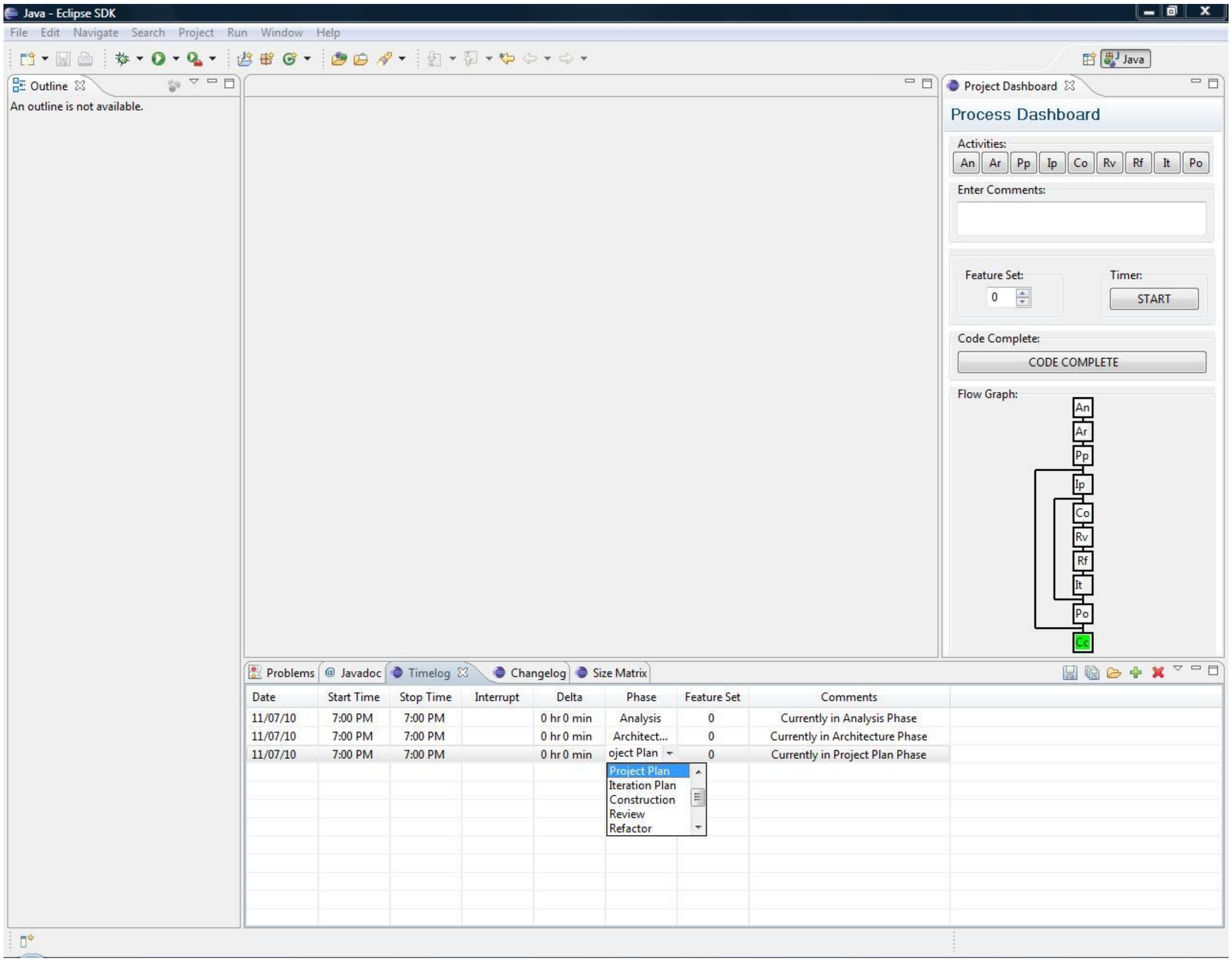
Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase
11/07/10	7:00 PM	7:01 PM			Iteration ...	0	Currently in Iteration Plan Phase
11/07/10	7:01 PM	7:01 PM			Construct...	0	Iteration 2
11/08/10	8:23 AM	9:25 AM			Analysis	0	Starting

Screen shot 7: Inserting a new row into the Timelog



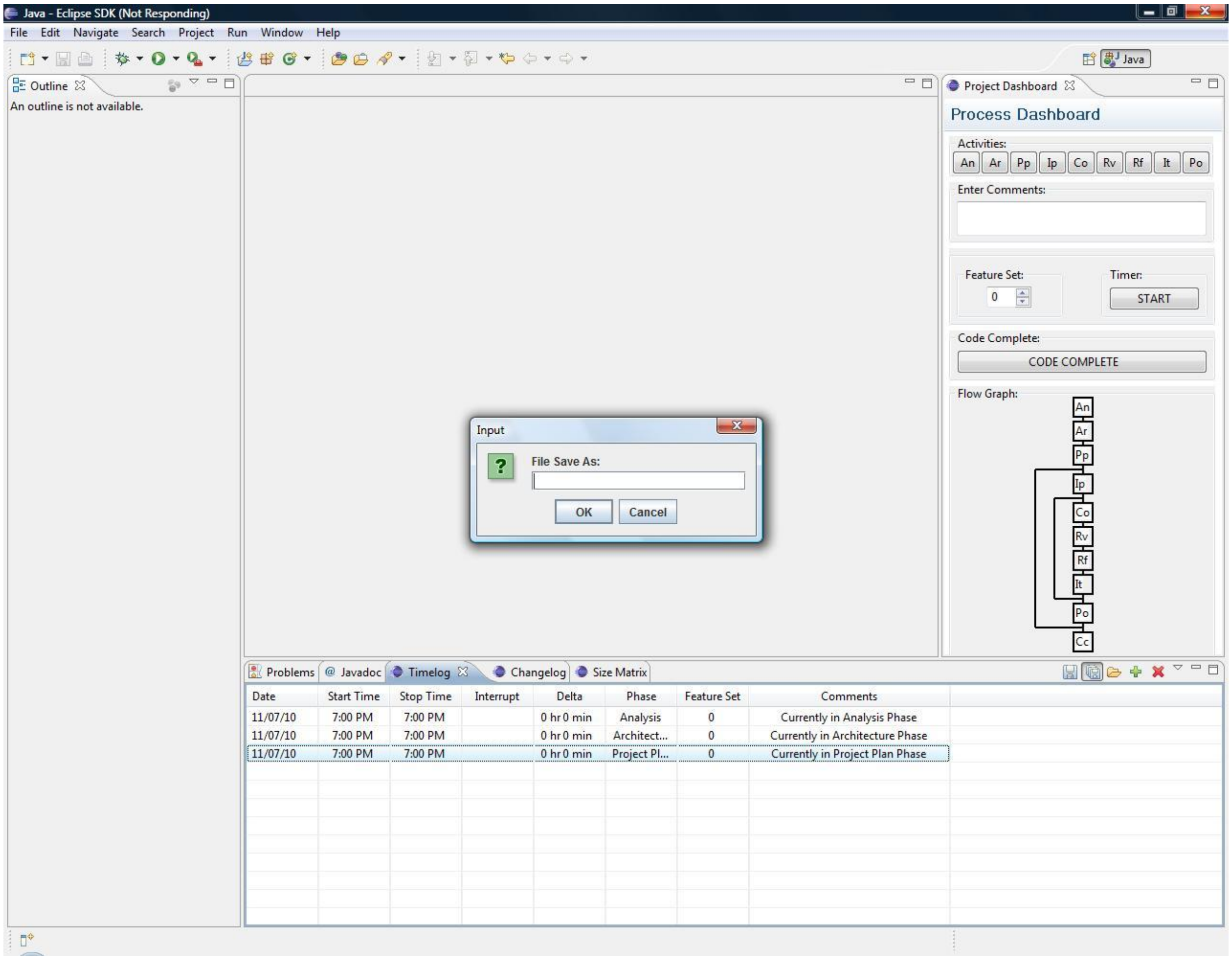
Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase
11/07/10	7:01 PM	7:01 PM		0 hr 0 min	Construct...	0	Iteration 2
11/08/10	8:23 AM	9:25 AM			alysis	0	Starting

Screen shot 8: Deleting a row from the timelog



Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	oject Plan	0	Currently in Project Plan Phase

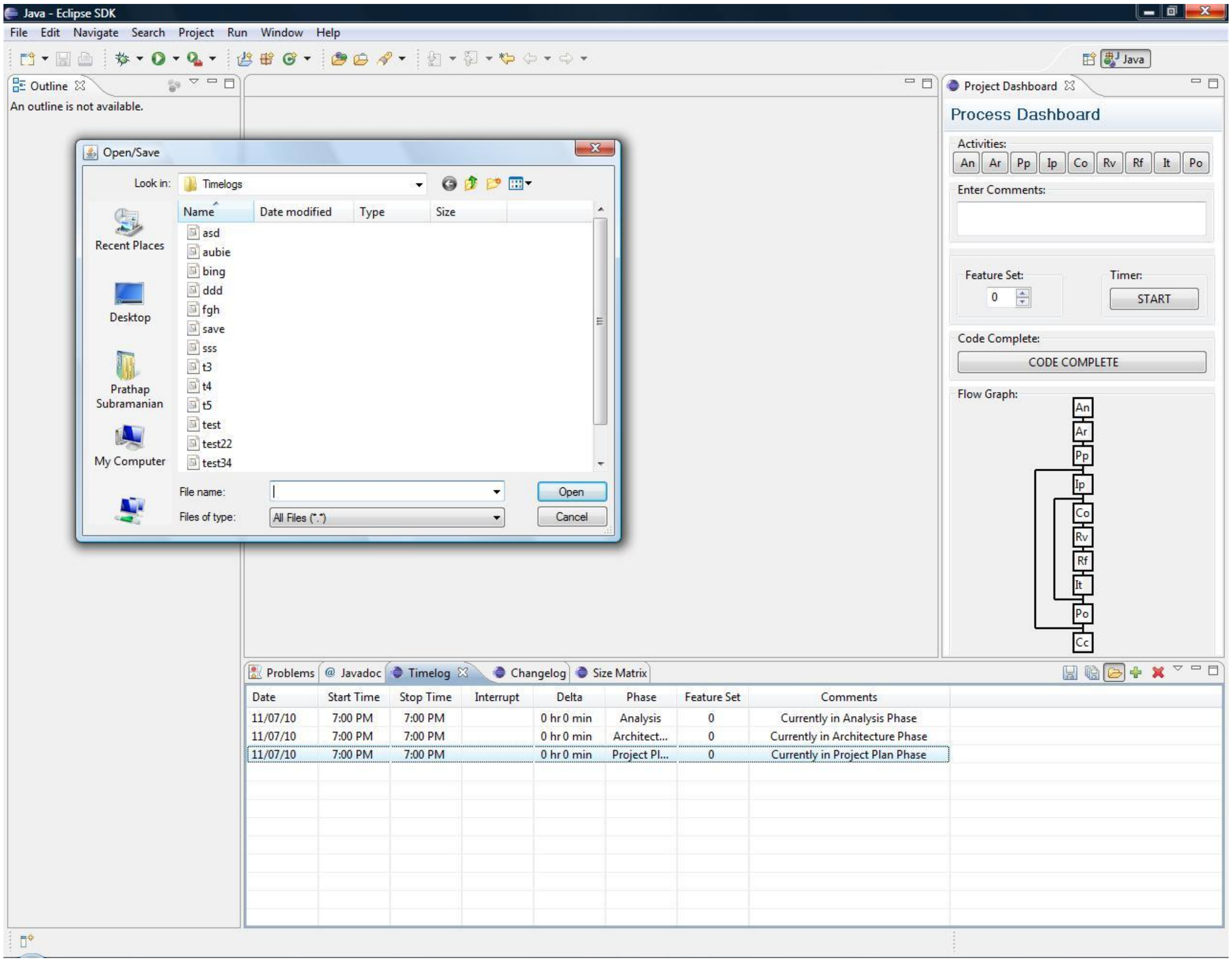
Screen shot 9: Display of timelog phases in a combo box



An outline is not available.

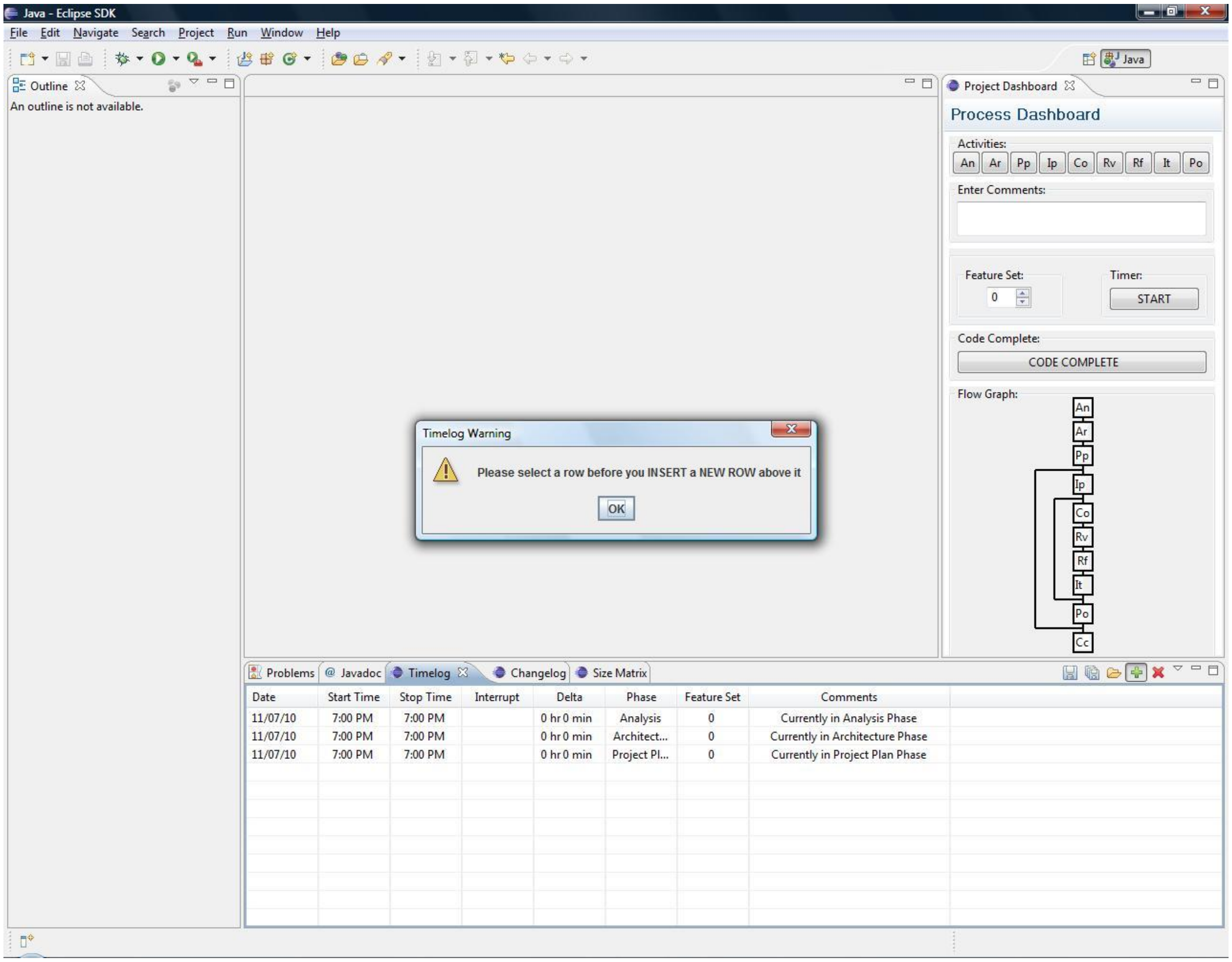
Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase

Screen shot 10: File SaveAs dialog



Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase

Screen shot 11: File Open dialog



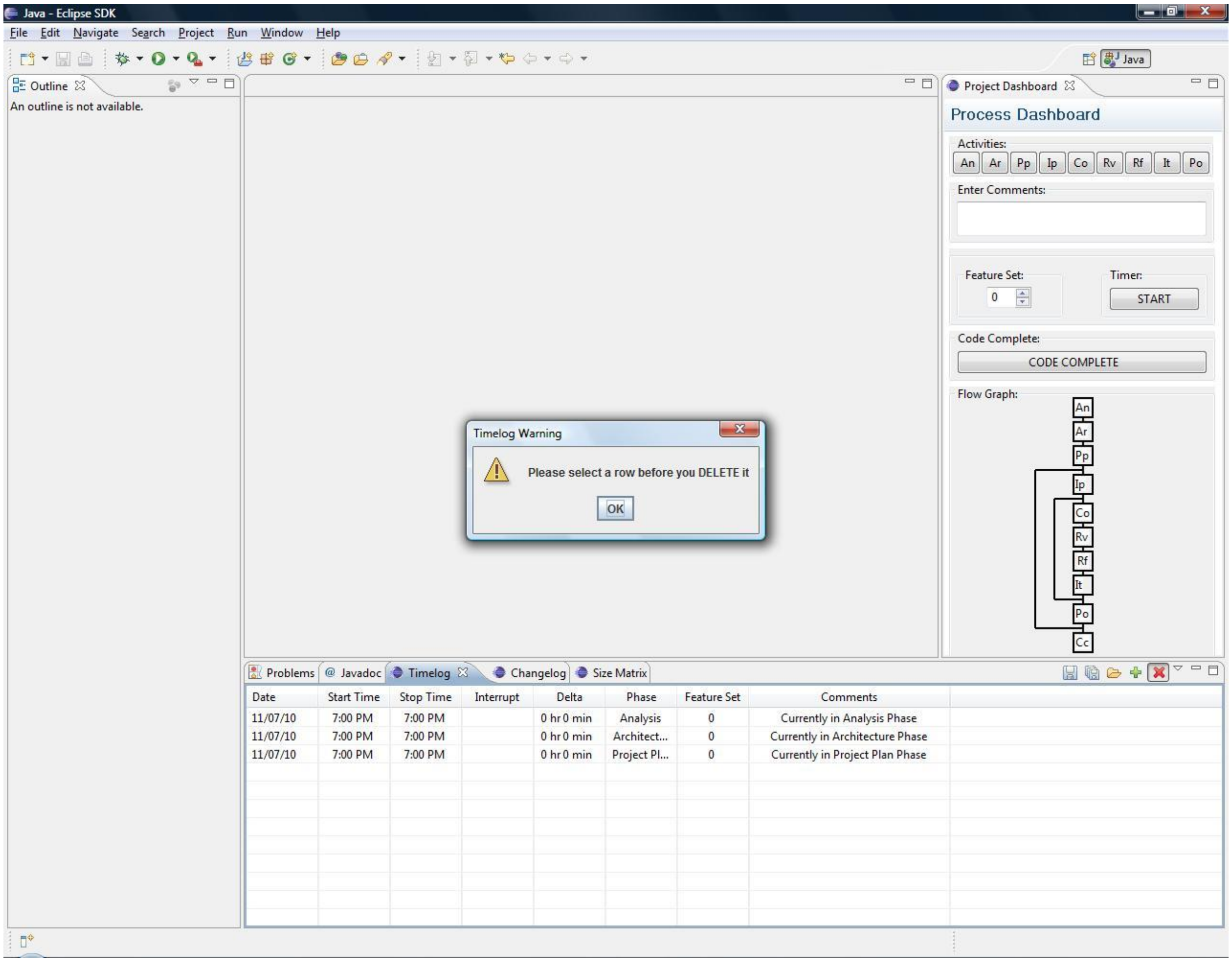
An outline is not available.

Timelog Warning

Please select a row before you INSERT a NEW ROW above it

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase

Screen shot 12: Timelog and Changelog validation



An outline is not available.

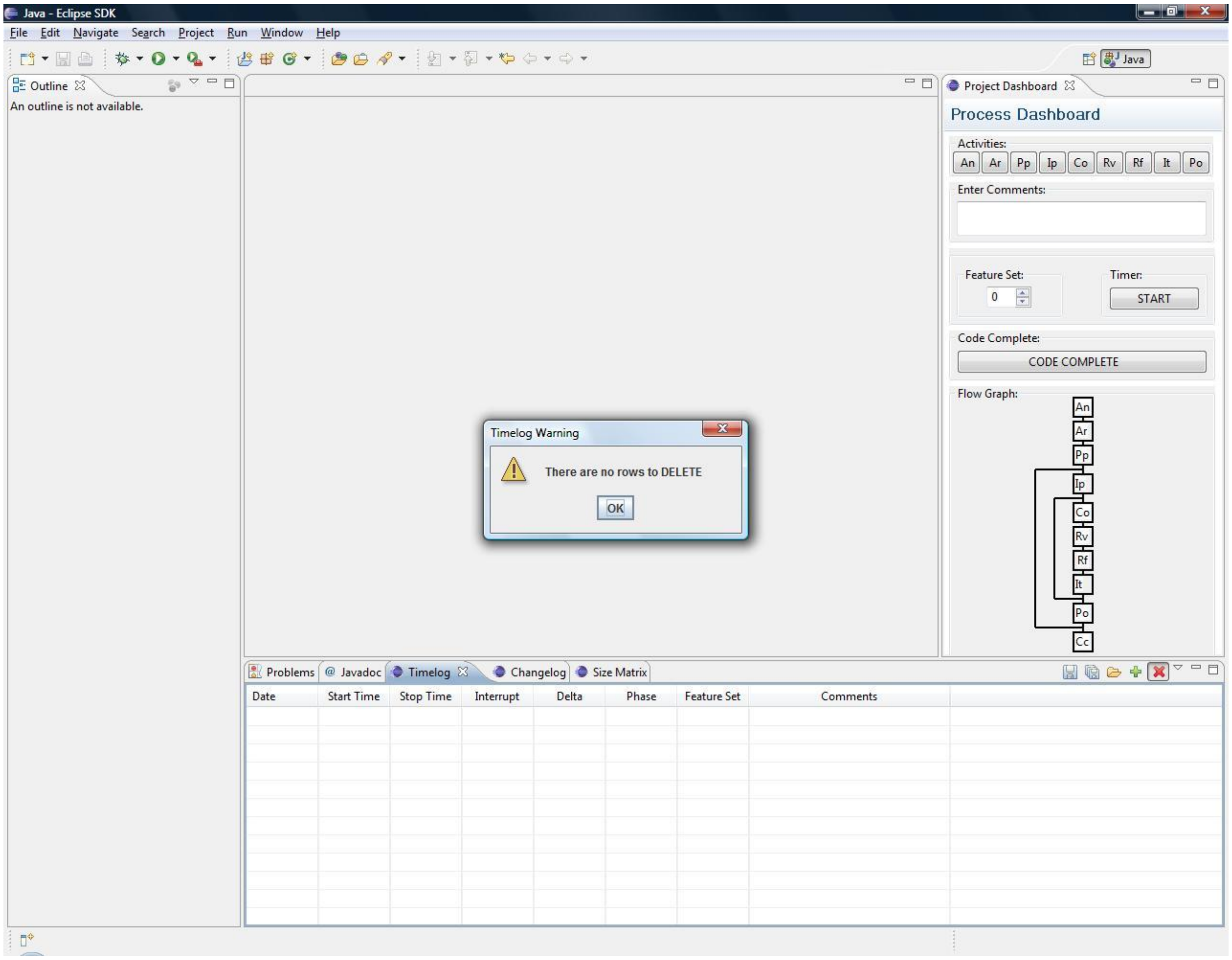
Feature Set: 0

Timelog Warning

Please select a row before you DELETE it

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Analysis	0	Currently in Analysis Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Architect...	0	Currently in Architecture Phase
11/07/10	7:00 PM	7:00 PM		0 hr 0 min	Project Pl...	0	Currently in Project Plan Phase

Screen shot 13: Timelog and Changelog validation



An outline is not available.

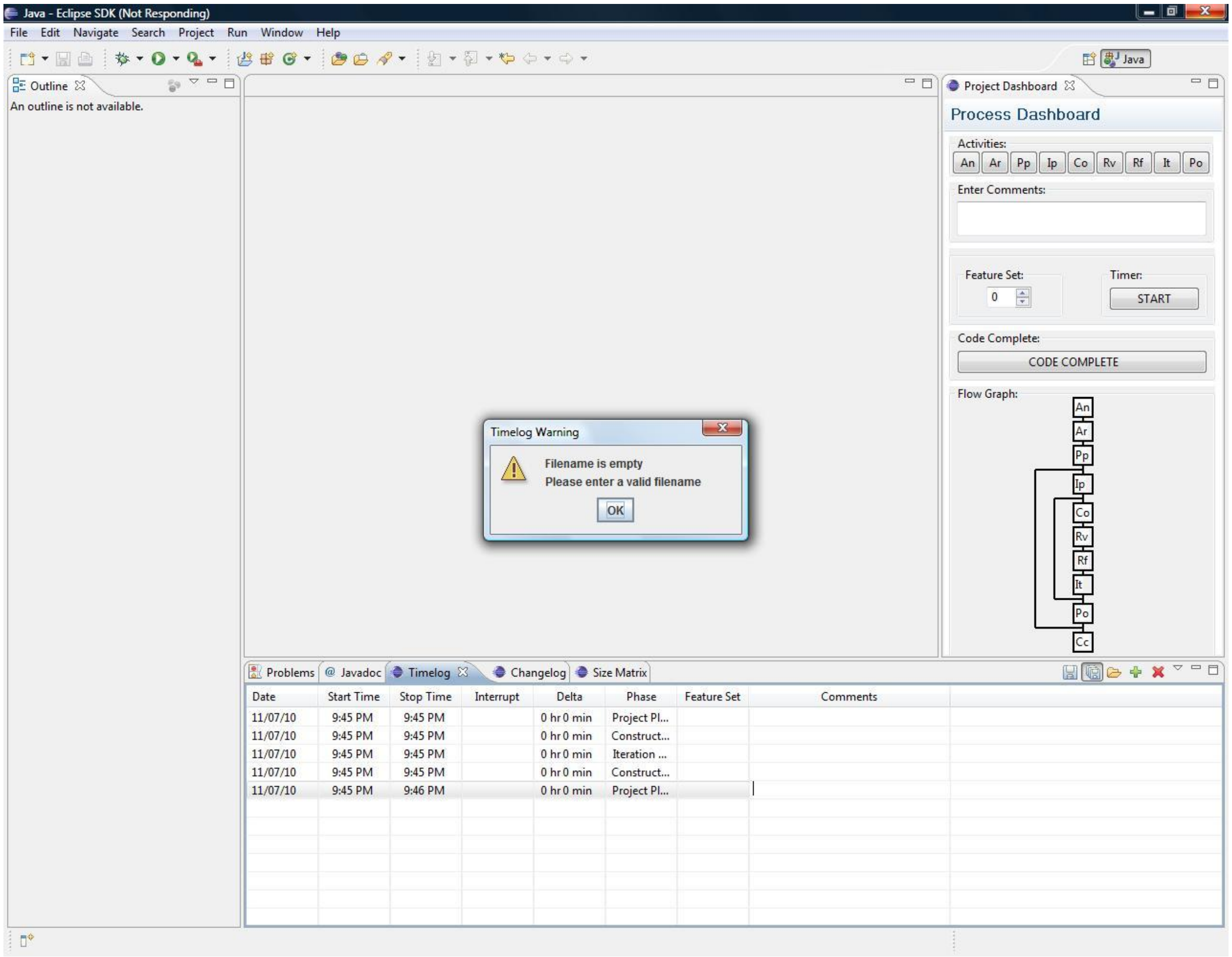
Feature Set: 0

Timelog Warning

There are no rows to DELETE

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments

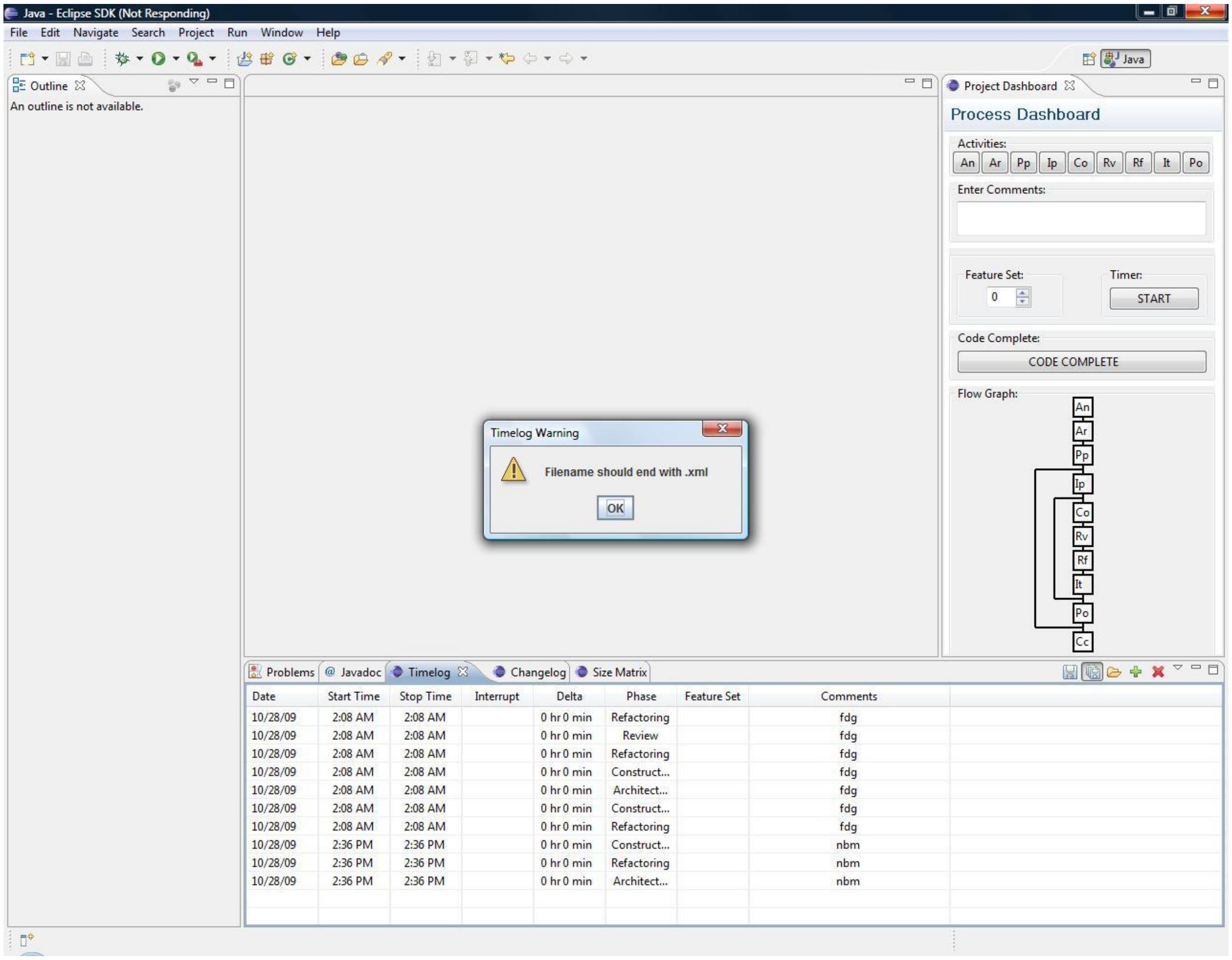
Screen shot 14: Timelog and Changelog validation



Filename is empty
Please enter a valid filename

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/07/10	9:45 PM	9:45 PM		0 hr 0 min	Project Pl...		
11/07/10	9:45 PM	9:45 PM		0 hr 0 min	Construct...		
11/07/10	9:45 PM	9:45 PM		0 hr 0 min	Iteration ...		
11/07/10	9:45 PM	9:45 PM		0 hr 0 min	Construct...		
11/07/10	9:45 PM	9:46 PM		0 hr 0 min	Project Pl...		

Screen shot 15: Timelog SaveAs validation



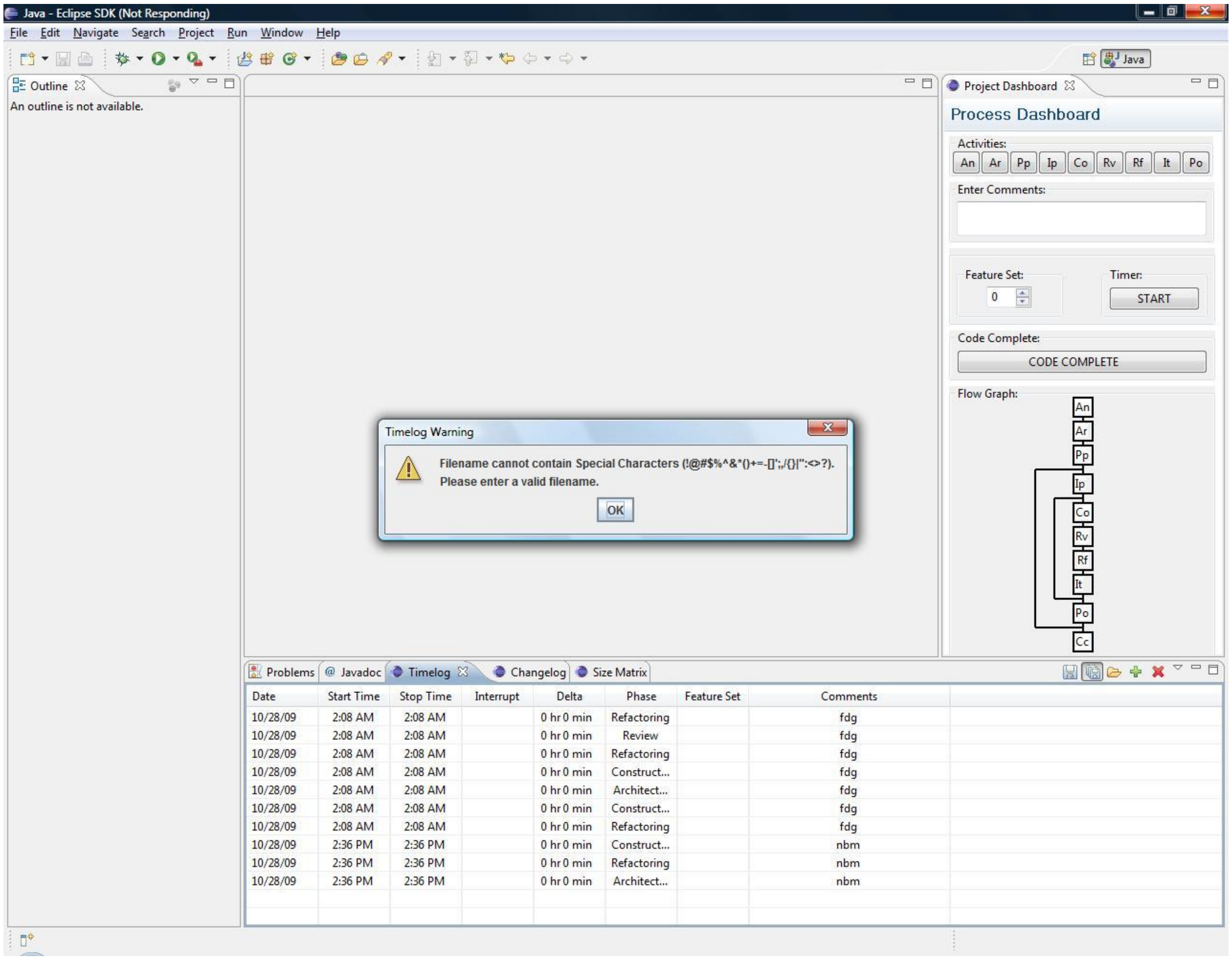
An outline is not available.

Timelog Warning

Filename should end with .xml

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Refactoring		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Review		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Refactoring		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Construct...		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Architect...		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Construct...		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Refactoring		fdg
10/28/09	2:36 PM	2:36 PM		0 hr 0 min	Construct...		nbm
10/28/09	2:36 PM	2:36 PM		0 hr 0 min	Refactoring		nbm
10/28/09	2:36 PM	2:36 PM		0 hr 0 min	Architect...		nbm

Screen shot 16: Timelog SaveAs validation

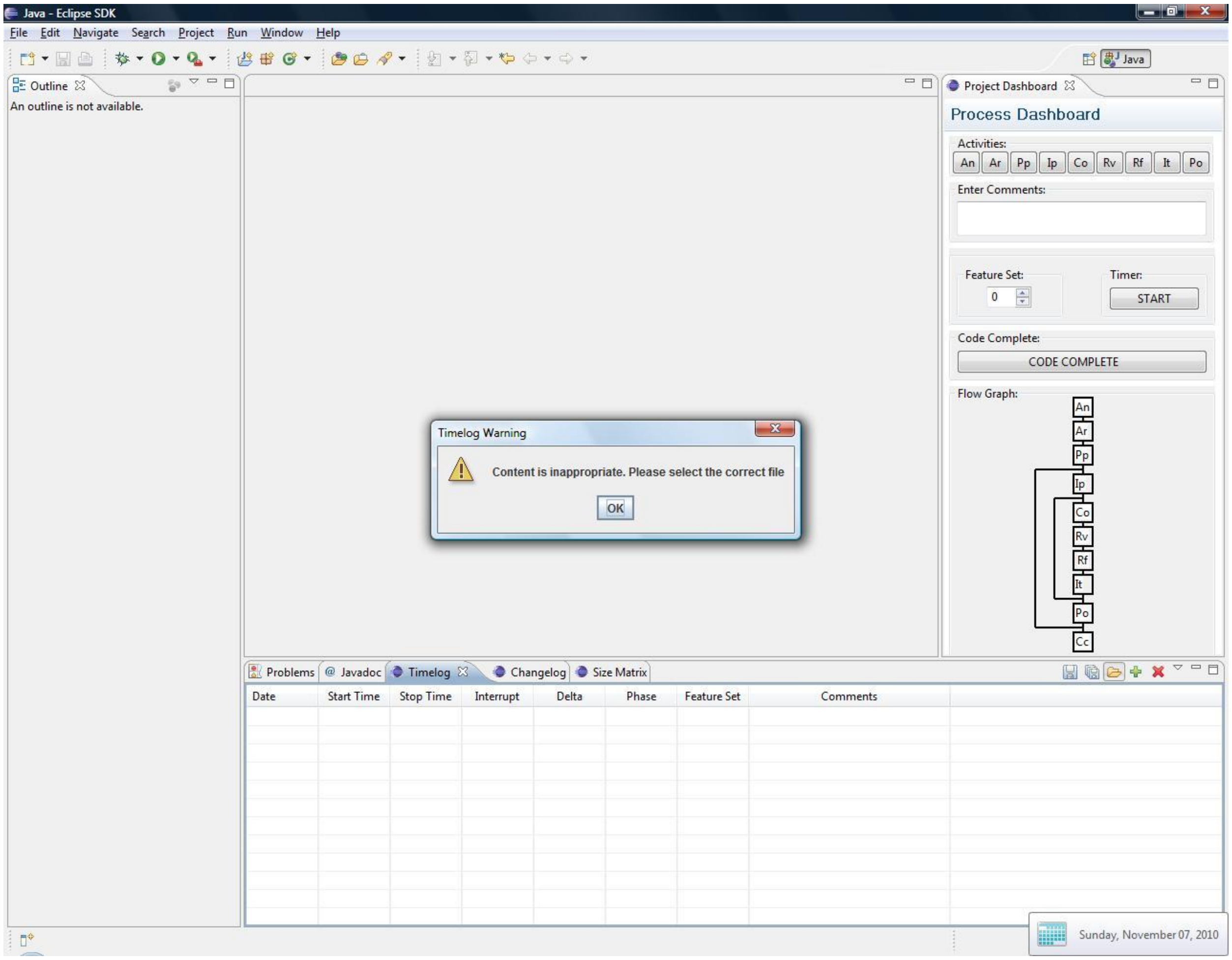


Feature Set: 0

Filename cannot contain Special Characters (!@#$%^&*()+=-[]';,/{}|":<>?).
Please enter a valid filename.

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Refactoring		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Review		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Refactoring		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Construct...		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Architect...		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Construct...		fdg
10/28/09	2:08 AM	2:08 AM		0 hr 0 min	Refactoring		fdg
10/28/09	2:36 PM	2:36 PM		0 hr 0 min	Construct...		nbm
10/28/09	2:36 PM	2:36 PM		0 hr 0 min	Refactoring		nbm
10/28/09	2:36 PM	2:36 PM		0 hr 0 min	Architect...		nbm

Screen shot 17: Timelog SaveAs validation



An outline is not available.

Feature Set: 0

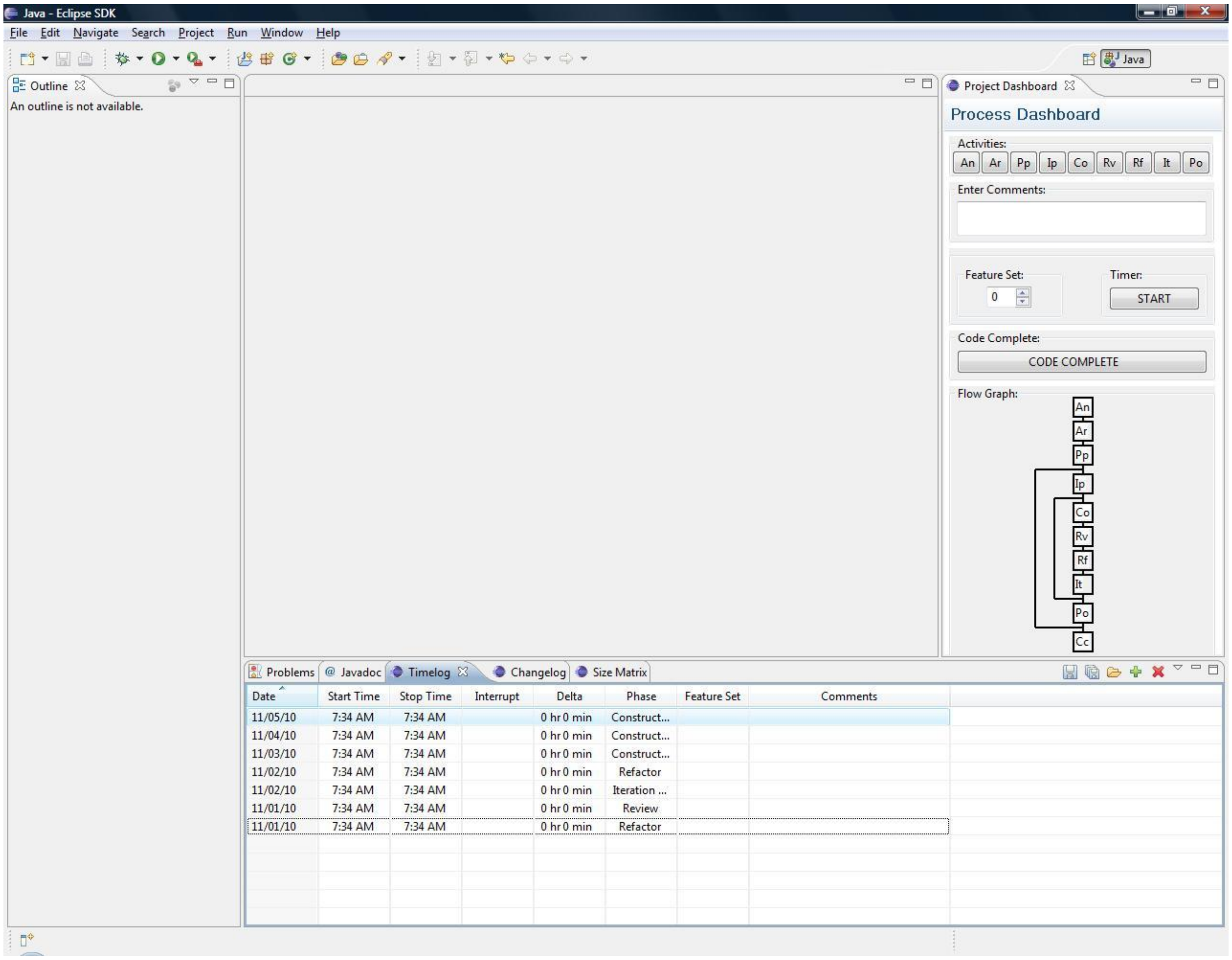
Timelog Warning

Content is inappropriate. Please select the correct file

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments

Sunday, November 07, 2010

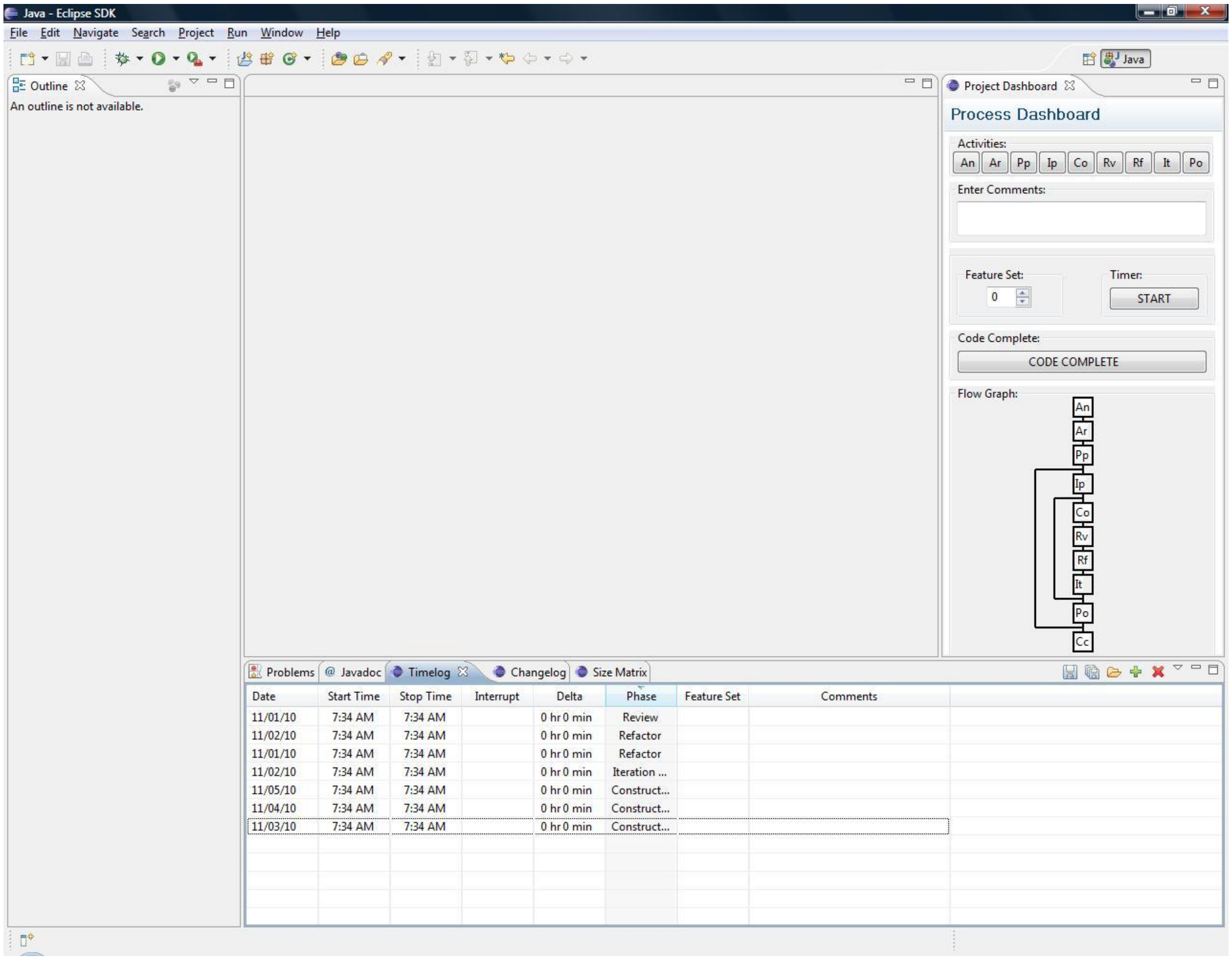
Screen shot 18: Timelog File Open validation



Feature Set: 0

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/05/10	7:34 AM	7:34 AM		0 hr 0 min	Construct...		
11/04/10	7:34 AM	7:34 AM		0 hr 0 min	Construct...		
11/03/10	7:34 AM	7:34 AM		0 hr 0 min	Construct...		
11/02/10	7:34 AM	7:34 AM		0 hr 0 min	Refactor		
11/02/10	7:34 AM	7:34 AM		0 hr 0 min	Iteration ...		
11/01/10	7:34 AM	7:34 AM		0 hr 0 min	Review		
11/01/10	7:34 AM	7:34 AM		0 hr 0 min	Refactor		

Screen shot 19: Sort by date



An outline is not available.

Feature Set: 0

Date	Start Time	Stop Time	Interrupt	Delta	Phase	Feature Set	Comments
11/01/10	7:34 AM	7:34 AM		0 hr 0 min	Review		
11/02/10	7:34 AM	7:34 AM		0 hr 0 min	Refactor		
11/01/10	7:34 AM	7:34 AM		0 hr 0 min	Refactor		
11/02/10	7:34 AM	7:34 AM		0 hr 0 min	Iteration ...		
11/05/10	7:34 AM	7:34 AM		0 hr 0 min	Construct...		
11/04/10	7:34 AM	7:34 AM		0 hr 0 min	Construct...		
11/03/10	7:34 AM	7:34 AM		0 hr 0 min	Construct...		

Screen shot 20: Sort by phase

Appendix 3 (Source code)

Timelog.java

```
/* Timelog class to represent the timelog data members
 * with getter and setter methods
 */

package org.example.tableviewer.views;

import java.beans.PropertyChangeSupport;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "timelog")
public class Timelog {

      private int counter;

      private boolean completed = true;
      private String date = "";
      private String starttime = "";
      private String stoptime = "";
      private String interrupt = "";
      private String delta = "";
      private String phase = "";
      private String featureset = "";
      private String comments = "";

      private long calcStarttime = 0;
      private long calcStoptime = 0;

      private int intStartStopButPressFlag = 0;
      private int intPhaseAfterStartButPressFlag = 0;
      private int intAddlnPhaseRowsFlag = 0;


      private PropertyChangeSupport propertyChangeSupport = new
PropertyChangeSupport(
                  this);


      public Timelog(int counter) {
            this.counter = counter;
      }

      public Timelog(){
            super();
            this.completed = true;
            this.date = "";
            this.starttime = "";
```

```
        this.stoptime = "";
        this.interrupt = "";
        this.delta = "";
        this.phase = "";
        this.featureset = "";
        this.comments = "";

        this.intStartStopButPressFlag = 0;
        this.intPhaseAfterStartButPressFlag = 0;
        this.intAddlnPhaseRowsFlag = 0;
}

public String toString() {
        return "Item " + this.counter;
}

public boolean isCompleted() {
        return completed;
}

public String getDate() {
        return date;
}

public String getStarttime() {
        return starttime;
}

public String getStoptime() {
        return stoptime;
}

public String getInterrupt() {
        return interrupt;
}

public String getDelta() {
        return delta;
}

public String getPhase() {
        return phase;
}


public String getFeatureset() {
        return featureset;
}


public String getComments() {
        return comments;
}

public long getCalcStarttime() {
        return calcStarttime;
}
```

```
public long getCalcStoptime() {
        return calcStoptime;
}

public int getIntStartStopButPressFlag() {
        return intStartStopButPressFlag;
}

public int getIntPhaseAfterStartButPressFlag() {
        return intPhaseAfterStartButPressFlag;
}

public int getIntAddlnPhaseRowsFlag() {
        return intAddlnPhaseRowsFlag;
}


public void setCompleted(boolean completed) {
        this.completed = completed;
}

public void setDate(String date) {
        propertyChangeSupport.firePropertyChange("Date", this.date,
        this.date = date);
}

public void setStarttime(String starttime) {
        propertyChangeSupport.firePropertyChange("Starttime",
        this.starttime,this.starttime = starttime);
}

public void setStoptime(String stoptime) {
        propertyChangeSupport.firePropertyChange("Stoptime",
        this.stoptime,this.stoptime = stoptime);
}

public void setInterrupt(String interrupt) {
        propertyChangeSupport.firePropertyChange("Interrupt",
        this.interrupt,this.interrupt = interrupt);
}

public void setDelta(String delta) {
        propertyChangeSupport.firePropertyChange("Delta", this.delta,
        this.delta = delta);
}

public void setPhase(String phase) {
        propertyChangeSupport.firePropertyChange("Phase", this.phase,
        this.phase = phase);
}


public void setFeatureSet(String featureset) {
        propertyChangeSupport.firePropertyChange("FeatureSet",
        this.featureset,this.featureset = featureset);
}
public void setComments(String comments) {
```

```
            propertyChangeSupport.firePropertyChange("Comments",
            this.comments,this.comments = comments);
      }

      public void setCalcStarttime(long calcStarttime) {
            this.calcStarttime = calcStarttime;
      }

      public void setCalcStoptime(long calcStoptime) {
            this.calcStoptime = calcStoptime;
      }

      public void setIntStartStopButPressFlag(int intStartStopButPressFlag) {
            this.intStartStopButPressFlag = intStartStopButPressFlag;
      }

      public void setIntPhaseAfterStartButPressFlag(int
                  intPhaseAfterStartButPressFlag) {
            this.intPhaseAfterStartButPressFlag =
                  intPhaseAfterStartButPressFlag;
      }

      public void setIntAddlnPhaseRowsFlag(int intAddlnPhaseRowsFlag) {
            this.intAddlnPhaseRowsFlag = intAddlnPhaseRowsFlag;
      }

    }
```

TimelogSorter.java

```
/*
 * TimelogSorter class to sort the Date,Phase and Featureset columns
 */
package org.example.tableviewer.views;

import org.eclipse.jface.viewers.Viewer;
import org.eclipse.jface.viewers.ViewerSorter;

public class TimelogSorter extends ViewerSorter{

      private int propertyIndex;
      private static final int DESCENDING = 1;

      private int direction = DESCENDING;

      public TimelogSorter() {
            this.propertyIndex = 0;
            direction = DESCENDING;
      }

      public void setColumn(int column) {

            if (column == this.propertyIndex) {
                  //Same column as last sort; toggle the direction
                  direction = 1 - direction;
            } else {
```

```
            //New column; do an ascending sort
            this.propertyIndex = column;
            direction = DESCENDING;
        }


    }


    public int compare(Viewer viewer, Object e1, Object e2) {
        Timelog tObj1 = (Timelog) e1;
        Timelog tObj2 = (Timelog) e2;
        int rc = 0;
        switch (propertyIndex) {
        case 0:
            rc = tObj1.getDate().compareTo(tObj2.getDate());
            break;
        case 1:
            break;
        case 2:
            break;
        case 3:
            break;
        case 4:
            break;
        case 5:
            rc = tObj1.getPhase().compareTo(tObj2.getPhase());
            break;
        case 6:
            rc =
            tObj1.getFeatureset().compareTo(tObj2.getFeatureset());
            break;
        case 7:
            break;
        default:
            rc = 0;
        }

        // If descending order, flip the direction
        if (direction == DESCENDING) {
            rc = -rc;
        }

        return rc;
    }


}
```

TimelogStore.java

```
/*
```

```
 * TimelogStore to save the Timelog as xml
 */
package org.example.tableviewer.views;

import java.util.ArrayList;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

//This statement means that class "TimelogStore.java" is the root-element
@XmlRootElement(namespace = "org.example.tableviewer.views")
public class TimelogStore {

//XmLElementWrapper generates a wrapper element around XML representation
      @XmlElementWrapper(name = "timelogs")
      //XmlElement sets the name of the entities
      @XmlElement(name = "timelog")
      private ArrayList<Timelog> timelogs;
      private String name;


      public void setTimelogList(ArrayList<Timelog> timelogList) {
            this.timelogs = timelogList;
      }

      public ArrayList<Timelog> getTimelogsList() {
            return timelogs;
      }

      public String getName() {
            return name;
      }

      public void setName(String name) {
            this.name = name;
      }
}
```

TimelogView.java

```
/*
 * TimelogView demonstrates the timelog plug-in view
 */

package org.example.tableviewer.views;

import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.Writer;
import java.util.ArrayList;
import java.util.Properties;

import javax.swing.JOptionPane;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import javax.xml.bind.UnmarshalException;
import javax.xml.bind.Unmarshaller;

import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.ui.part.*;
import org.eclipse.jface.viewers.*;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.graphics.*;
import org.eclipse.jface.action.*;
import org.eclipse.ui.*;
import org.eclipse.swt.widgets.Menu;
import org.eclipse.swt.*;



/**
 * This class demonstrates how to plug-in a new
 * workbench view.The view is connected to the model using
 * a content provider.The view uses a label provider to define
 * how model objects should be presented in the view. Each
 * view can present the same model objects using
 * different labels and icons, if needed. Alternatively,
 * a single label provider can be shared between views
 * in order to ensure that objects of the same type are
 * presented in the same way everywhere.
 *
 */

public class TimelogView extends ViewPart implements ISelectionListener{


      //The ID of the view as specified by the extension.
      public static final String ID =
                              "org.example.tableviewer.views.SampleView";

      final String PROPFILE=
      "C:/Users/pzs0011/workspace/org.example.TableViewer/attributes.properti
      es";
      Properties myProp;


      private TableViewer viewer;
      private TimelogSorter timelogSorter;

      private ArrayList<Timelog> timelogs;

      private String glbFilename = "";
      private Action save;
      private Action saveas;
      private Action open;
      private Action insert;
      private Action delete;

      private String TIMELOG_XML = "";
      private String TIMELOGOPEN_XML = "";

      private String[] columnNames;

      private int insertAtIndex;

      static int intAddlnPhaseRowsFlag = 0;

      /*
       * The content provider class is responsible for
       * providing objects to the view. It can wrap
       * existing objects in adapters or simply return
       * objects as-is. These objects may be sensitive
       * to the current input of the view, or ignore
       * it and always show the same content
       * (like Task List, for example).
       */

      class ViewContentProvider implements IStructuredContentProvider {
            public void inputChanged(Viewer v, Object oldInput, Object
            newInput) {
            }
            public void dispose() {
            }
            public Object[] getElements(Object parent) {
                  return new String[] { "One", "Two", "Three" };
            }
      }

      class ViewLabelProvider extends LabelProvider implements
            ITableLabelProvider {
            public String getColumnText(Object obj, int index) {
                  return getText(obj);
            }
            public Image getColumnImage(Object obj, int index) {
                  return getImage(obj);
            }
            public Image getImage(Object obj) {
                  return PlatformUI.getWorkbench().

            getSharedImages().getImage(ISharedImages.IMG_OBJ_ELEMENT);
            }
      }
      class NameSorter extends ViewerSorter {
      }

      //Constructor
      public TimelogView() {
      }
      /**
       * This is a callback that will allow us
       * to create the viewer and initialize it.
       */
      public void createPartControl(Composite parent) {

            getSite().getPage().addSelectionListener(this);


            viewer = new TableViewer(parent, SWT.MULTI | SWT.H_SCROLL |
                  SWT.V_SCROLL | SWT.FULL_SELECTION);
            final Table table = viewer.getTable();


            table.setHeaderVisible(true);
            table.setLinesVisible(true);

            timelogSorter = new TimelogSorter();


            columnNames = new String[] {
                  "Date", "Start Time", "Stop Time", "Interrupt", "Delta",
                  "Phase", "Feature Set", "Comments"};
            int[] columnWidths = new int[] { 75,75,75,75,75,75,75,210};
            int[] columnAlignments = new int[] {
      SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.C
      ENTER,SWT.CENTER};


            for(int i=0; i<columnNames.length;i++)
            {
                  final int index = i;

                  TableViewerColumn viewerColumn = new
                                    TableViewerColumn(viewer, SWT.NONE);
                  final TableColumn tableColumn = viewerColumn.getColumn();

                  viewerColumn.getColumn().setText(columnNames[i]);
                  viewerColumn.getColumn().setWidth(columnWidths[i]);
                  viewerColumn.getColumn().setAlignment(columnAlignments[i]);
                  viewerColumn.getColumn().setResizable(true);
                  viewerColumn.getColumn().setMoveable(true);

                  // Setting the right sorter
                  tableColumn.addSelectionListener(new SelectionAdapter() {
                        public void widgetSelected(SelectionEvent e) {

                        //Get the content for the viewer,setInput will set
                        //the no. of timelog records(rows) for the sort
                        timelogs = new ArrayList<Timelog>();
                        int totalCount = viewer.getTable().getItemCount();

                        for(int c = 0; c < totalCount; c++)
                        {
                              Timelog tObj = (Timelog)viewer.getElementAt(c);
                              timelogs.add(tObj);
                        }
                        viewer.setInput(timelogs);

                        // Set the sorter for the table
```

```
                        viewer.setSorter(timelogSorter);

                        if(index == 0 || index == 5 || index == 6)
                        {
                              timelogSorter.setColumn(index);
                              int dir = viewer.getTable().getSortDirection();
                              if (viewer.getTable().getSortColumn() ==
                                    tableColumn) {
                                    dir = dir == SWT.UP ? SWT.DOWN : SWT.UP;
                              } else {
                                    dir = SWT.DOWN;
                              }
                              viewer.getTable().setSortDirection(dir);
                              viewer.getTable().setSortColumn(tableColumn);
                              viewer.refresh();
                        }
                     }
                  });

      //Enable editing support by calling the class cellEditingSupport
      viewerColumn.setEditingSupport(new CellEditingTimelog(viewer, i));
      }

      viewer.setLabelProvider(new PersonTableLabelProvider());
      viewer.setContentProvider(new ArrayContentProvider());

      //Create the help context id for the viewer's control

      PlatformUI.getWorkbench().getHelpSystem().setHelp(viewer.getControl(),
      "org.example.TableViewer.viewer");
            makeActions();
            hookContextMenu();
            contributeToActionBars();

 }


public class PersonTableLabelProvider extends LabelProvider implements
      ITableLabelProvider
      {

            @Override
            public Image getColumnImage(Object element, int columnIndex) {
                  return null;
            }

            public String getColumnText(Object element, int index)
            {
                  Timelog timelog = (Timelog) element;
                  switch(index){
                  case 0:
                        return timelog.getDate();
                  case 1:
                        return timelog.getStarttime();
                  case 2:
                        return timelog.getStoptime();
                  case 3:
```

```
                  return timelog.getInterrupt();
            case 4:
                  return timelog.getDelta();
            case 5:
                  return timelog.getPhase();
            case 6:
                  return timelog.getFeatureset();
            case 7:
                  return timelog.getComments();
            default:
                  return "unknown" + index;
            }
      }

}




private void hookContextMenu() {
      MenuManager menuMgr = new MenuManager("#PopupMenu");
      menuMgr.setRemoveAllWhenShown(true);
      menuMgr.addMenuListener(new IMenuListener() {
            public void menuAboutToShow(IMenuManager manager) {
                  TimelogView.this.fillContextMenu(manager);
            }
      });
      Menu menu = menuMgr.createContextMenu(viewer.getControl());
      viewer.getControl().setMenu(menu);
      getSite().registerContextMenu(menuMgr, viewer);
}

private void contributeToActionBars() {
      IActionBars bars = getViewSite().getActionBars();
      fillLocalPullDown(bars.getMenuManager());
      fillLocalToolBar(bars.getToolBarManager());
}

private void fillLocalPullDown(IMenuManager manager) {
      manager.add(insert);
      manager.add(new Separator());
      manager.add(delete);
}

private void fillContextMenu(IMenuManager manager) {
      manager.add(insert);
      manager.add(delete);
      // Other plug-ins can contribute there actions here
      manager.add(new
Separator(IWorkbenchActionConstants.MB_ADDITIONS));
}

private void fillLocalToolBar(IToolBarManager manager) {
      manager.add(save);
      manager.add(saveas);
      manager.add(open);
      manager.add(insert);
      manager.add(delete);
}


private void makeActions() {

      save = new Action() {
            public void run() {

               try
               {
                     if(glbFilename.equals(""))
                     {
                           saveas.run();
                     }
                     else
                     {
                      //Load the property file before use
                        myProp = loadProperties(PROPFILE);

                     //Read SaveAs path from properties file
                   TIMELOG_XML =
                  myProp.getProperty("timelog_save_path")+glbFilename;


            //Get the content for the viewer,setInput will set the no.
            of timelog records(rows) for the sort
            timelogs = new ArrayList<Timelog>();
            int totalCount = viewer.getTable().getItemCount();

            for(int c = 0; c < totalCount; c++)
            {
                  Timelog tObj = (Timelog)viewer.getElementAt(c);
                  timelogs.add(tObj);
            }

            //create timelogstore, assigning timelog
            TimelogStore timelogstore = new TimelogStore();
            timelogstore.setName(glbFilename+" Timelog");
            timelogstore.setTimelogList(timelogs);

            // create JAXB context and instantiate marshaller
            try
            {
                  JAXBContext context =
                        JAXBContext.newInstance(TimelogStore.class);
                  Marshaller m = context.createMarshaller();

                  m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                                Boolean.TRUE);

                  Writer w = null;
                  try {
                          w = new FileWriter(TIMELOG_XML);
                          m.marshal(timelogstore, w);
                      } finally {
                  try {
```

```
                                w.close();
                            } catch (Exception e) {}
                              }
                              }catch(Exception e)
                              {
                                    e.printStackTrace();

                                    if(e.getClass().toString().equals("class
                                    java.io.FileNotFoundException"))
                                    {
                                    //Java Swing dialog box
                                    JOptionPane.showMessageDialog(null,
                                    "Unable to save timelog as xml. \nSystem
cannot find the PATH specified. \nPlease change the settings appropriately",
                                                "Timelog Warning",
                                    JOptionPane.WARNING_MESSAGE);
                                    }
                              }

                           }

               }
               catch(Exception e1)
               {
                   e1.printStackTrace();
               }

            }
            };
            save.setText("Save");
            save.setToolTipText("Save Timelog");

      save.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_ETOOL_SAVE_EDIT));


            saveas = new Action() {
                  public void run() {

                        String filename = "";
                        String specialCharacters = "!@#$%^&*()+=-
                        []\';,/{}|\":<>?";


                        //Java Swing dialog box
                        filename = JOptionPane.showInputDialog(null,
                            "File Save As:");

                        if(filename.equals(""))
                        {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                            "Filename is empty \nPlease enter a valid
                              filename","Timelog Warning",
                                  JOptionPane.WARNING_MESSAGE);

                              saveas.run();
                        }else if(!filename.endsWith(".xml"))
                        {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                                  "Filename should end with .xml",
                                  "Timelog Warning",
                                  JOptionPane.WARNING_MESSAGE);

                              saveas.run();
                        }else if(filename.endsWith(".xml") ||
                              filename.endsWith(".XML"))
                        {

                              for (int i=0; i < filename.length(); i++)
                              {

      if(specialCharacters.indexOf(filename.charAt(i)) != -1)
                                    {
                                    //Java Swing dialog box
                                    JOptionPane.showMessageDialog(null,
                                    "Filename cannot contain Special
Characters (!@#$%^&*()+=-[]\';,/{}|\":<>?). \nPlease enter a valid
filename.","Timelog Warning",JOptionPane.WARNING_MESSAGE);

                                          saveas.run();
                                    }
                              }

                              filename = filename.trim();
                              glbFilename = filename;

                              //Load the property file before use
                              myProp = loadProperties(PROPFILE);

                              //Read SaveAs path from properties file.
                              TIMELOG_XML =
                        myProp.getProperty("timelog_saveas_path")+filename;

                        //Get the content for the viewer,setInput will set
the no. of timelog records(rows) for the sort
                        timelogs = new ArrayList<Timelog>();
                        int totalCount = viewer.getTable().getItemCount();

                        for(int c = 0; c < totalCount; c++)
                        {
                              Timelog tObj = (Timelog)viewer.getElementAt(c);
                              timelogs.add(tObj);
                        }

                      //create timelogstore, assigning timelog
                        TimelogStore timelogstore = new TimelogStore();
                        timelogstore.setName(filename+" Timelog");
                        timelogstore.setTimelogList(timelogs);
                        // create JAXB context and instantiate marshaller
                        try
                        {
                              JAXBContext context =
                              JAXBContext.newInstance(TimelogStore.class);
                              Marshaller m = context.createMarshaller();
                              m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                                            Boolean.TRUE);

                              Writer w = null;
                              try {
                                    w = new FileWriter(TIMELOG_XML);
                                    m.marshal(timelogstore, w);
                              } finally {
                                    try {
                                          w.close();
                                    } catch (Exception e) {}
                              }
                        }catch(Exception e)
                        {
                              e.printStackTrace();

                              if(e.getClass().toString().equals("class
                                          java.io.FileNotFoundException"))
                              {
                                    //Java Swing dialog box
                                    JOptionPane.showMessageDialog(null,
                                    "Unable to save timelog as xml. \nSystem
cannot find the PATH specified. \nPlease change the settings appropriately",
                        "Timelog Warning",JOptionPane.WARNING_MESSAGE);
                              }
                        }
                  }
                  }
            };
            saveas.setText("Save As");
            saveas.setToolTipText("SaveAs Timelog");

      saveas.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_ETOOL_SAVEALL_EDIT));


            open = new Action() {

                  public void run()
                  {
                        String directoryFile = new
FileBrowse().getDirectoryFile();
                        TIMELOGOPEN_XML = directoryFile;

                        try
                        {
                              if( !TIMELOGOPEN_XML.equals("") &&
TIMELOGOPEN_XML.endsWith("xml") || TIMELOGOPEN_XML.endsWith("XML") )
                              {
                              //Remove current elements before inserting
                              int tableCount =
viewer.getTable().getItemCount();
                              viewer.getTable().remove(0,tableCount-1);
                              try
                              {
                                    JAXBContext context =
                              JAXBContext.newInstance(TimelogStore.class);
                              Unmarshaller um = context.createUnmarshaller();

                              TimelogStore timelogstore2 = (TimelogStore)
                        um.unmarshal(new FileReader(TIMELOGOPEN_XML));

                        if(timelogstore2 instanceof TimelogStore)
                                          {
                        for (int i = 0; i <
                  timelogstore2.getTimelogsList().toArray().length; i++)
                        {
                              Timelog tObj =
                              (Timelog)(timelogstore2.getTimelogsList().get(i
                              ));
                              viewer.insert(tObj,i);
                        }
                        }
                  }
                        catch(UnmarshalException ex)
                        {
                              JOptionPane.showMessageDialog(null,
                            "Content is inappropriate. Please select the
correct file", "Timelog Warning",JOptionPane.WARNING_MESSAGE);
                        }

                              }
                              else
                              {
                                    //Java Swing dialog box
                                    JOptionPane.showMessageDialog(null,
                                    "Content is inappropriate. Please select
the correct file", "Timelog Warning",JOptionPane.WARNING_MESSAGE);
                              }

                        }catch(Exception e)
                        {
                              e.printStackTrace();
                        }
                  }
            };
            open.setText("Open");
            open.setToolTipText("Open Timelog");

      open.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_OBJ_FOLDER));


            insert = new Action() {
                  public void run() {
                        //Insert row logic goes here
                        IStructuredSelection selection =
                              (IStructuredSelection)viewer.getSelection();
                  if ( (viewer.getTable().getItemCount() >= 1) &&
                        (selection.isEmpty()) )
                        {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                            "Please select a row before you INSERT a NEW ROW
                            above it","Timelog Warning",
                            JOptionPane.WARNING_MESSAGE);
                        }
                        else if(viewer.getTable().getItemCount() == 0)
                        {
                              Timelog tObj = new Timelog();
                              viewer.insert(tObj, 0);

                        }
                        else if(!(selection.isEmpty()) &&
                        (viewer.getTable().getItemCount() >= 1))
                        {

                              for(Object o:selection.toList())
                              {
                                    Timelog tgObj = new Timelog();
                                    o = tgObj;
                                    if( o instanceof Timelog)
                                    {
                                    viewer.insert(o,
                                    viewer.getTable().getSelectionIndex());
                                    }
                              }
                        }

                  }
            };
            insert.setText("Insert");
            insert.setToolTipText("Insert new row");

      insert.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_OBJ_ADD));



            delete = new Action() {
                  public void run() {
                  //Delete row logic goes here
                  IStructuredSelection selection =
                  (IStructuredSelection)viewer.getSelection();

                  if ( (viewer.getTable().getItemCount() >= 1) &&
                        (selection.isEmpty()) )
                        {
                        //Java Swing dialog box
                        JOptionPane.showMessageDialog(null,
                        "Please select a row before you DELETE it",
                        "Timelog Warning",JOptionPane.WARNING_MESSAGE);
                        }else if(viewer.getTable().getItemCount() == 0)
                        {
```

```
                  //Java Swing dialog box
                  JOptionPane.showMessageDialog(null,
                  "There are no rows to DELETE",
                  "Timelog Warning",
                  JOptionPane.WARNING_MESSAGE);

                  }else if(!(selection.isEmpty()))
                  {
                  int index = viewer.getTable().getSelectionIndex();
                  viewer.getTable().remove(index);
                  }
            }
      };
      delete.setText("Delete");
      delete.setToolTipText("Delete row");

delete.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_ETOOL_DELETE));

}


/**
 * Passing the focus request to the viewer's control.
 */
public void setFocus() {
      viewer.getControl().setFocus();
}


@Override
public void selectionChanged(IWorkbenchPart part, ISelection selection)
{

      if(!(selection instanceof IStructuredSelection))
            return;
      IStructuredSelection ss = (IStructuredSelection) selection;

      Object o = ss.getFirstElement();
      if( o instanceof Timelog)
      {

      //Switching Phases after starting the START button
      if( ((Timelog) o).getIntStartStopButPressFlag() == 1 &&
      ((Timelog) o).getIntPhaseAfterStartButPressFlag() == 1 &&
      ((Timelog) o).getIntAddlnPhaseRowsFlag() == 0)
      {
            viewer.getTable().remove(insertAtIndex);
            viewer.insert(o, insertAtIndex);
      }
      else if( ((Timelog) o).getIntStartStopButPressFlag() == 1 &&
      ((Timelog) o).getIntPhaseAfterStartButPressFlag() == 1 &&
      ((Timelog) o).getIntAddlnPhaseRowsFlag() == 1)
      {
            insertAtIndex = viewer.getTable().getItemCount();
            viewer.insert(o, insertAtIndex);
      }
```

```
            //START - Insert
            else if(((Timelog) o).getIntStartStopButPressFlag() == 1)
            {
                  insertAtIndex = viewer.getTable().getItemCount();
                  viewer.insert(o, insertAtIndex);
            }
            //STOP - Remove and Insert
            else if(((Timelog) o).getIntStartStopButPressFlag() == 0)
            {
                  viewer.getTable().remove(insertAtIndex);
                  viewer.insert(o, insertAtIndex);
            }
         }
      }


      //Function to load the properties file
      private static Properties loadProperties(String fileName)
      {
            InputStream propsFile;
        Properties tempProp = new Properties();

        try
        {
            propsFile = new FileInputStream(fileName);
            tempProp.load(propsFile);
            propsFile.close();
        }catch (IOException ioe) {
            System.out.println("I/O Exception.");
            ioe.printStackTrace();
            System.exit(0);
        }

        return tempProp;

    }

}
```

cellEditingTimelog.java

```
/*
 * CellEditingTimelog to support editing the timelog rows
 */

package org.example.tableviewer.views;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
import org.eclipse.jface.viewers.CellEditor;
import org.eclipse.jface.viewers.ColumnViewer;
import org.eclipse.jface.viewers.ComboBoxCellEditor;
import org.eclipse.jface.viewers.EditingSupport;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.jface.viewers.TextCellEditor;


public class CellEditingTimelog extends EditingSupport {
      private CellEditor editor;
      private int column;

      final String PROPFILE=
"C:/Users/pzs0011/workspace/org.example.TableViewer/attributes.properties";
      Properties myProp;

      public CellEditingTimelog(ColumnViewer viewer, int column) {
            super(viewer);

            //Load the property file before use
            myProp = loadProperties(PROPFILE);

          //Please Note: The phases/activities are read from PROPERTY FILE
            (Key=Value) pair.
            String[] activities = new String[10];
            activities[0] = myProp.getProperty("activity_1");
            activities[1] = myProp.getProperty("activity_2");
            activities[2] = myProp.getProperty("activity_3");
            activities[3] = myProp.getProperty("activity_4");
            activities[4] = myProp.getProperty("activity_5");
            activities[5] = myProp.getProperty("activity_6");
            activities[6] = myProp.getProperty("activity_7");
            activities[7] = myProp.getProperty("activity_8");
            activities[8] = myProp.getProperty("activity_9");
            activities[9] = myProp.getProperty("activity_10");


            //Create the correct editor based on the column index
            switch (column)
            {
            case 0:
                  editor = new TextCellEditor(((TableViewer)
                        viewer).getTable());
                        break;
                  case 1:
                  editor = new TextCellEditor(((TableViewer)
                        viewer).getTable());
                        break;
                  case 2:
                  editor = new TextCellEditor(((TableViewer)
                        viewer).getTable());
                        break;
                  case 3:
                  editor = new TextCellEditor(((TableViewer)
                        viewer).getTable());
                        break;
                  case 4:
                  editor = new TextCellEditor(((TableViewer)
                        viewer).getTable());
                        break;
            case 5:
            editor = new ComboBoxCellEditor(((TableViewer)
                viewer).getTable(),activities);
                break;
            case 6:
            editor = new TextCellEditor(((TableViewer)
                viewer).getTable());
                break;
            case 7:
            editor = new TextCellEditor(((TableViewer)
                viewer).getTable());
                break;
            default:
            editor = new TextCellEditor(((TableViewer)
                viewer).getTable());
        }

        this.column = column;

    }

    @Override
    protected boolean canEdit(Object element) {
        return true;
    }

    @Override
    protected CellEditor getCellEditor(Object element) {
        return editor;
    }

    @Override
    protected Object getValue(Object element) {
        Timelog person = (Timelog) element;

        //Load the property file before use
        myProp = loadProperties(PROPFILE);

        switch (this.column) {
        case 0:
            return person.getDate();
        case 1:
            return person.getStarttime();
        case 2:
            return person.getStoptime();
        case 3:
            return person.getInterrupt();
        case 4:
            return person.getDelta();
        case 5:

            System.out.println("GET PHASE"+person.getPhase());

            //Please Note: The phases/activities are read from PROPERTY
            FILE (Key=Value) pair.
            if (person.getPhase().equals(
                   myProp.getProperty("activity_1")) ) {
```

```
            return 0;
        }
        else if (person.getPhase().equals(
             myProp.getProperty("activity_2")) ) {
            return 1;
        }
        else if(person.getPhase().equals(
            myProp.getProperty("activity_3")) ) {
            return 2;
        }
        else if(person.getPhase().equals(
            myProp.getProperty("activity_4")) ) {
            return 3;
        }
        else if(person.getPhase().equals(
            myProp.getProperty("activity_5")) ) {
            return 4;
        }
        else if(person.getPhase().equals(
            myProp.getProperty("activity_6")) ) {
            return 5;
        }
        else if(person.getPhase().equals(
            myProp.getProperty("activity_7")) ) {
            return 6;
        }
        else if(person.getPhase().equals(
            myProp.getProperty("activity_8")) ) {
            return 7;
        }
        else if(person.getPhase().equals(
            myProp.getProperty("activity_9")) ) {
            return 8;
        }
        else if(person.getPhase().equals(
             myProp.getProperty("activity_10")) ) {
            return 9;
        }

    case 6:
        return person.getFeatureset();
    case 7:
        return person.getComments();
    default:
        break;
    }


    return null;
}

@Override
protected void setValue(Object element, Object value) {
    Timelog pers = (Timelog) element;

    //Load the property file before use
    myProp = loadProperties(PROPFILE);
```

```
        switch (this.column) {
        case 0:
                pers.setDate(String.valueOf(value));
                break;
        case 1:
                pers.setStarttime(String.valueOf(value));
                break;
        case 2:
                pers.setStoptime(String.valueOf(value));
                break;
        case 3:
                pers.setInterrupt(String.valueOf(value));
                break;
        case 4:
                pers.setDelta(String.valueOf(value));
                break;
        case 5:
                //Please Note: The phases/activities are read from PROPERTY
                FILE (Key=Value) pair.
                if (((Integer) value) == 0) {
                        pers.setPhase( myProp.getProperty("activity_1") );
                } else if (((Integer) value) == 1) {
                        pers.setPhase( myProp.getProperty("activity_2") );
                } else if(((Integer) value) == 2) {
                        pers.setPhase( myProp.getProperty("activity_3") );
                } else if(((Integer) value) == 3) {
                        pers.setPhase( myProp.getProperty("activity_4") );
                } else if(((Integer) value) == 4) {
                        pers.setPhase( myProp.getProperty("activity_5") );
                } else if(((Integer) value) == 5) {
                        pers.setPhase( myProp.getProperty("activity_6") );
                } else if(((Integer) value) == 6) {
                        pers.setPhase( myProp.getProperty("activity_7") );
                } else if(((Integer) value) == 7) {
                        pers.setPhase( myProp.getProperty("activity_8") );
                } else if(((Integer) value) == 8) {
                        pers.setPhase( myProp.getProperty("activity_9") );
                } else if(((Integer) value) == 9) {
                        pers.setPhase( myProp.getProperty("activity_10") );
                }
                break;
        case 6:
                pers.setFeatureSet(String.valueOf(value));
                break;
        case 7:
                pers.setComments(String.valueOf(value));
                break;
        default:
                break;
        }
        getViewer().update(element, null);
}

//Function to load the properties file
private static Properties loadProperties(String fileName)
{
```

```
            InputStream propsFile;
        Properties tempProp = new Properties();

        try
        {
            propsFile = new FileInputStream(fileName);
            tempProp.load(propsFile);
            propsFile.close();
        }catch (IOException ioe) {
            System.out.println("I/O Exception.");
            ioe.printStackTrace();
            System.exit(0);
        }

        return tempProp;

    }

}
```

Changelog.java

```
/*
 * Changelog class to represent the changelog data members
 * with getter and setter methods
 */
package org.example.tableviewer.views;

import java.beans.PropertyChangeSupport;

import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "changelog")
public class Changelog {

      private String number = "";
      private String date = "";
      private String type = "";
      private String injectactivity = "";
      private String injectfeatureset = "";
      private String removeactivity = "";
      private String removefeatureset = "";
      private String fixtime = "";
      private String fixreference = "";
      private String description = "";
      private PropertyChangeSupport propertyChangeSupport = new
PropertyChangeSupport(
                  this);

      public Changelog(){
            super();
            this.number = "";
```

```
        this.date = "";
        this.type = "";
        this.injectactivity = "";
        this.injectfeatureset = "";
        this.removeactivity = "";
        this.removefeatureset = "";
        this.fixtime = "";
        this.fixreference = "";
        this.description = "";

}


public String getNumber() {
        return number;
}

public String getDate() {
        return date;
}

public String getType() {
        return type;
}

public String getInjectactivity() {
        return injectactivity;
}

public String getInjectfeatureset() {
        return injectfeatureset;
}

public String getRemoveactivity() {
        return removeactivity;
}

public String getRemovefeatureset() {
        return removefeatureset;
}

public String getFixtime() {
        return fixtime;
}

public String getFixreference() {
        return fixreference;
}

public String getDescription() {
        return description;
}

public void setNumber(String number) {
        propertyChangeSupport.firePropertyChange("Number", this.number,
                        this.number = number);
}
```

```
      public void setDate(String date) {
            propertyChangeSupport.firePropertyChange("Date", this.date,
                        this.date = date);
      }

      public void setType(String type) {
            propertyChangeSupport.firePropertyChange("Type", this.type,
                        this.type = type);
      }

      public void setInjectactivity(String injectactivity) {
            propertyChangeSupport.firePropertyChange("Inject Activity",
            this.injectactivity,this.injectactivity = injectactivity);
      }

      public void setInjectfeatureset(String injectfeatureset) {
            propertyChangeSupport.firePropertyChange("Inject FeatureSet",
            this.injectfeatureset,this.injectfeatureset = injectfeatureset);
      }

      public void setRemoveactivity(String removeactivity) {
            propertyChangeSupport.firePropertyChange("Remove Activity",
            this.removeactivity,this.removeactivity = removeactivity);
      }

      public void setRemovefeatureset(String removefeatureset) {
            propertyChangeSupport.firePropertyChange("Remove FeatureSet",
            this.removefeatureset,this.removefeatureset = removefeatureset);
      }

      public void setFixtime(String fixtime) {
            propertyChangeSupport.firePropertyChange("Fix Time",
            this.fixtime,this.fixtime = fixtime);
      }

      public void setFixreference(String fixreference) {
            propertyChangeSupport.firePropertyChange("Fix Reference",
            this.fixreference,this.fixreference = fixreference);
      }

      public void setDescription(String description) {
            propertyChangeSupport.firePropertyChange("Description",
            this.description,this.description = description);
      }
}
```

ChangelogSorter.java

```
/*
 * ChangelogSorter class to sort the columns: Number, Date, Type,
 * Inject Activity, Inject Featureset, Remove Activity,
 * Remove Featureset, Fix Time
 */
```

```
package org.example.tableviewer.views;

import org.eclipse.jface.viewers.Viewer;
import org.eclipse.jface.viewers.ViewerSorter;

public class ChangelogSorter extends ViewerSorter{

      private int propertyIndex;
      private static final int DESCENDING = 1;

      private int direction = DESCENDING;

      public ChangelogSorter() {
            this.propertyIndex = 0;
            direction = DESCENDING;
      }

      public void setColumn(int column) {

            if (column == this.propertyIndex) {
                  // Same column as last sort; toggle the direction
                  direction = 1 - direction;
            } else {
                  // New column; do an ascending sort
                  this.propertyIndex = column;
                  direction = DESCENDING;
            }

      }

      public int compare(Viewer viewer, Object e1, Object e2) {
            Changelog cObj1 = (Changelog) e1;
            Changelog cObj2 = (Changelog) e2;
            int rc = 0;
            switch (propertyIndex) {
            case 0:
                  rc = cObj1.getNumber().compareTo(cObj2.getNumber());
                  break;
            case 1:
                  rc = cObj1.getDate().compareTo(cObj2.getDate());
                  break;
            case 2:
                  rc = cObj1.getType().compareTo(cObj2.getType());
                  break;
            case 3:
                  rc =
            cObj1.getInjectactivity().compareTo(cObj2.getInjectactivity());
                  break;
            case 4:
                  rc =
            cObj1.getInjectfeatureset().compareTo(cObj2.getInjectfeatureset()
            );
                  break;
            case 5:
                  rc =
            cObj1.getRemoveactivity().compareTo(cObj2.getRemoveactivity());
                  break;
            case 6:
                  rc =
            cObj1.getRemovefeatureset().compareTo(cObj2.getRemovefeatureset()
            );
                  break;
            case 7:
                  rc = cObj1.getFixtime().compareTo(cObj2.getFixtime());
                  break;
            case 8:

                  break;
            case 9:

                  break;
            default:
                  rc = 0;
            }

            // If descending order, flip the direction
            if (direction == DESCENDING) {
                  rc = -rc;
            }

            return rc;
      }


}
```

ChangelogStore.java

```
/*
 * ChangelogStore to save the changelog as xml
 */
package org.example.tableviewer.views;

import java.util.ArrayList;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

//This statement means that class "ChangelogStore.java" is the root-element
@XmlRootElement(namespace = "org.example.tableviewer.views")
public class ChangelogStore {

      // XmLElementWrapper generates a wrapper element around XML
      representation
      @XmlElementWrapper(name = "changelogs")
      // XmlElement sets the name of the entities
      @XmlElement(name = "changelog")
      private ArrayList<Changelog> changelogs;
      private String name;
	public void setChangelogList(ArrayList<Changelog> changelogList) {
		this.changelogs = changelogList;
	}

	public ArrayList<Changelog> getChangelogsList() {
		return changelogs;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}
```

ChangelogView.java

```
/*
 * ChangelogView demonstrates the Changelog plug-in view
 */

package org.example.tableviewer.views;

import java.io.FileInputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.Writer;
import java.util.ArrayList;
import java.util.Properties;

import javax.swing.JOptionPane;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Marshaller;
import javax.xml.bind.UnmarshalException;
import javax.xml.bind.Unmarshaller;

import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Table;
import org.eclipse.swt.widgets.TableColumn;
import org.eclipse.ui.part.*;
import org.eclipse.jface.viewers.*;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.graphics.*;
import org.eclipse.jface.action.*;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.ui.*;
import org.eclipse.swt.widgets.Menu;
import org.eclipse.swt.*;
```

```
/**
 * This class demonstrates how to plug-in a new
 * workbench view.The view is connected to the model using
 * a content provider.The view uses a label provider to define
 * how model objects should be presented in the view. Each
 * view can present the same model objects using
 * different labels and icons, if needed. Alternatively,
 * a single label provider can be shared between views
 * in order to ensure that objects of the same type are
 * presented in the same way everywhere.
 *
 */
public class ChangelogView extends ViewPart implements ISelectionListener{

      /**
       * The ID of the view as specified by the extension.
       */
      public static final String ID =
            "org.example.tableviewer.views.SampleView";

      final String PROPFILE=
            "C:/Users/pzs0011/workspace/org.example.TableViewer/attributes.pr
      operties";
      Properties myProp;

      private TableViewer viewer;
      private ChangelogSorter changelogSorter;

      private ArrayList<Changelog> changelogs;

      private String glbFilename = "";
      private Action save;
      private Action saveas;
      private Action open;
      private Action insert;
      private Action delete;

      private String CHANGELOG_XML = "";
      private String CHANGELOGOPEN_XML = "";

      private String[] columnNames;

      static int intAddlnPhaseRowsFlag = 0;

      /*
       * The content provider class is responsible for
       * providing objects to the view. It can wrap
       * existing objects in adapters or simply return
       * objects as-is. These objects may be sensitive
       * to the current input of the view, or ignore
       * it and always show the same content
       * (like Task List, for example).
       */

      class ViewContentProvider implements IStructuredContentProvider {
```

```
      public void inputChanged(Viewer v, Object oldInput, Object
      newInput) {
      }
      public void dispose() {
      }
      public Object[] getElements(Object parent) {
            return new String[] { "One", "Two", "Three" };
      }
}
class ViewLabelProvider extends LabelProvider implements
      ITableLabelProvider {
      public String getColumnText(Object obj, int index) {
            return getText(obj);
      }
      public Image getColumnImage(Object obj, int index) {
            return getImage(obj);
      }
      public Image getImage(Object obj) {
            return PlatformUI.getWorkbench().

getSharedImages().getImage(ISharedImages.IMG_OBJ_ELEMENT);
      }
}
class NameSorter extends ViewerSorter {
}

//Constructor
public ChangelogView() {
}

/**
 * This is a callback that will allow us
 * to create the viewer and initialize it.
 */
public void createPartControl(Composite parent) {

      getSite().getPage().addSelectionListener(this);


      viewer = new TableViewer(parent, SWT.MULTI | SWT.H_SCROLL |
            SWT.V_SCROLL | SWT.FULL_SELECTION);
      final Table table = viewer.getTable();


      table.setHeaderVisible(true);
      table.setLinesVisible(true);

      changelogSorter = new ChangelogSorter();

      columnNames = new String[] {
            "Number", "Date", "Type", "Inject Activity", "Inject
            FeatureSet", "Remove Activity", "Remove FeatureSet", "Fix
            Time", "Fix Reference", "Description"};
      int[] columnWidths = new int[] {
            60,60,100,90,100,100,120,75,90,250};
      int[] columnAlignments = new int[] {
SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.C
```

```
ENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER};

for(int i=0; i<columnNames.length;i++)
{
      final int index = i;

      TableViewerColumn viewerColumn = new
            TableViewerColumn(viewer, SWT.NONE);
      final TableColumn tableColumn = viewerColumn.getColumn();

      viewerColumn.getColumn().setText(columnNames[i]);
      viewerColumn.getColumn().setWidth(columnWidths[i]);
      viewerColumn.getColumn().setAlignment(columnAlignments[i]);
      viewerColumn.getColumn().setResizable(true);
      viewerColumn.getColumn().setMoveable(true);

      // Setting the right sorter
      tableColumn.addSelectionListener(new SelectionAdapter() {
            public void widgetSelected(SelectionEvent e) {

      //Get the content for the viewer,setInput will set the no.
      of changelog records(rows) for the sort
      changelogs = new ArrayList<Changelog>();
      int totalCount = viewer.getTable().getItemCount();

      for(int c = 0; c < totalCount; c++)
      {
            Changelog cObj = (Changelog)viewer.getElementAt(c);
            changelogs.add(cObj);
      }
      viewer.setInput(changelogs);

      // Set the sorter for the table
      viewer.setSorter(changelogSorter);

      if(index == 0 || index == 1 || index == 2 || index == 3 ||
      index == 4 || index == 5 || index == 6 || index == 7)
      {
            changelogSorter.setColumn(index);
            int dir = viewer.getTable().getSortDirection();
            if (viewer.getTable().getSortColumn() == tableColumn)
            {
                  dir = dir == SWT.UP ? SWT.DOWN : SWT.UP;
            } else {
                  dir = SWT.DOWN;
            }
            viewer.getTable().setSortDirection(dir);

            viewer.getTable().setSortColumn(tableColumn);
            viewer.refresh();
            }
      }
});


// Enable editing support by calling the class
cellEditingChangelog
```

```
        viewerColumn.setEditingSupport(new CellEditingChangelog(viewer,
        i));

        }

        viewer.setLabelProvider(new ChangelogTableLabelProvider());
        viewer.setContentProvider(new ArrayContentProvider());

        // Create the help context id for the viewer's control

    PlatformUI.getWorkbench().getHelpSystem().setHelp(viewer.getControl(),
     "org.example.TableViewer.viewer");
        makeActions();
        hookContextMenu();
        hookDoubleClickAction();
        contributeToActionBars();

}


    private void hookDoubleClickAction() {

    }


    public class ChangelogTableLabelProvider extends LabelProvider
    implements ITableLabelProvider
    {

        public String getColumnText(Object element, int index)
        {
            Changelog changelog = (Changelog) element;
            switch(index){
            case 0:
                return changelog.getNumber();
            case 1:
                return changelog.getDate();
            case 2:
                return changelog.getType();
            case 3:
                return changelog.getInjectactivity();
            case 4:
                return changelog.getInjectfeatureset();
            case 5:
                return changelog.getRemoveactivity();
            case 6:
                return changelog.getRemovefeatureset();
            case 7:
                return changelog.getFixtime();
            case 8:
                return changelog.getFixreference();
            case 9:
                return changelog.getDescription();
            default:
                return "unknown" + index;
            }
        }
        @Override
        public Image getColumnImage(Object element, int columnIndex) {
            return null;
        }

    }


    private void hookContextMenu() {

        MenuManager menuMgr = new MenuManager("#PopupMenu");
        menuMgr.setRemoveAllWhenShown(true);
        menuMgr.addMenuListener(new IMenuListener() {
            public void menuAboutToShow(IMenuManager manager) {
                ChangelogView.this.fillContextMenu(manager);
            }
        });
        Menu menu = menuMgr.createContextMenu(viewer.getControl());
        viewer.getControl().setMenu(menu);
        getSite().registerContextMenu(menuMgr, viewer);
    }

    private void contributeToActionBars() {
        IActionBars bars = getViewSite().getActionBars();
        fillLocalPullDown(bars.getMenuManager());
        fillLocalToolBar(bars.getToolBarManager());
    }

    private void fillLocalPullDown(IMenuManager manager) {
        manager.add(insert);
        manager.add(new Separator());
        manager.add(delete);
    }

    private void fillContextMenu(IMenuManager manager) {
        manager.add(insert);
        manager.add(delete);
        // Other plug-ins can contribute there actions here
        manager.add(new
            Separator(IWorkbenchActionConstants.MB_ADDITIONS));
    }

    private void fillLocalToolBar(IToolBarManager manager) {
        manager.add(save);
        manager.add(saveas);
        manager.add(open);
        manager.add(insert);
        manager.add(delete);
    }


    private void makeActions() {


        save = new Action() {
            public void run() {
                    try
                    {
                          if(glbFilename.equals(""))
                          {
                                saveas.run();

                          }
                          else
                          {

                             //Load the property file before use
                             myProp = loadProperties(PROPFILE);

                             CHANGELOG_XML =
                myProp.getProperty("changelog_save_path")+glbFilename;

                //Get the content for the viewer,setInput will set the no.
                of Changelog records(rows) for the sort
                changelogs = new ArrayList<Changelog>();
                int totalCount = viewer.getTable().getItemCount();

                for(int c = 0; c < totalCount; c++)
                {
                      Changelog cObj = (Changelog)viewer.getElementAt(c);
                      changelogs.add(cObj);
                }

                //create Changelogstore, assigning Changelog
                ChangelogStore changelogstore = new ChangelogStore();
                changelogstore.setName(glbFilename+" Changelog");
                changelogstore.setChangelogList(changelogs);

                // create JAXB context and instantiate marshaller
                 try
                 {
                      JAXBContext context =
                      JAXBContext.newInstance(ChangelogStore.class);
                      Marshaller m = context.createMarshaller();

m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);

                      Writer w = null;
                      try {
                            w = new FileWriter(CHANGELOG_XML);
                            m.marshal(changelogstore, w);
                      } finally {
                            try {
                                  w.close();
                                  } catch (Exception e) {}
                                  }
                            }catch(Exception e)
                            {
                                  e.printStackTrace();

                                  if(e.getClass().toString().equals("class
                                  java.io.FileNotFoundException"))
```

```
                                    {
                                    //Java Swing dialog box
                                    JOptionPane.showMessageDialog(null,
                                    "Unable to save changelog as xml.
\nSystem cannot find the PATH specified. \nPlease change the settings
appropriately","Changelog Warning",JOptionPane.WARNING_MESSAGE);
                                    }
                              }

                           }

               }
                     catch(Exception e1)
                     {
                   e1.printStackTrace();
               }

                  }
            };
            save.setText("Save");
            save.setToolTipText("Save Changelog");

      save.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_ETOOL_SAVE_EDIT));


            saveas = new Action() {
                  public void run() {

                        String filename = "";
                        String specialCharacters = "!@#$%^&*()+=-
                              []\';,/{}|\":<>?";

                        //Java Swing dialog box
                        filename = JOptionPane.showInputDialog(null,
                            "File Save As:");

                        System.out.println("File Name is"+filename);

                        if(filename.equals(""))
                        {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                              "Filename is empty \nPlease enter a valid
filename","Changelog Warning",JOptionPane.WARNING_MESSAGE);

                              saveas.run();

                        }else if(!filename.endsWith(".xml"))
                        {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                                  "Filename should end with .xml",
                                  "Changelog Warning",
                                  JOptionPane.WARNING_MESSAGE);

                              saveas.run();
            }else if( filename.endsWith(".xml") ||
               filename.endsWith(".XML") )
            {

               for (int i=0; i < filename.length(); i++)
               {

   if(specialCharacters.indexOf(filename.charAt(i)) != -1)
                  {
                  //Java Swing dialog box
                  JOptionPane.showMessageDialog(null,
                  "Filename cannot contain Special
Characters (!@#$%^&*()+=-[]\';,/{}|\":<>?). \nPlease enter a valid
filename.","Changelog Warning",JOptionPane.WARNING_MESSAGE);

                     saveas.run();
                  }
               }

               filename = filename.trim();
               glbFilename = filename;

               //Load the property file before use
               myProp = loadProperties(PROPFILE);

               CHANGELOG_XML =
            myProp.getProperty("changelog_saveas_path")+filename;

            //Get the content for the viewer,setInput will set
the no. of changelog records(rows) for the sort
               changelogs = new ArrayList<Changelog>();
               int totalCount =
                  viewer.getTable().getItemCount();

               for(int c = 0; c < totalCount; c++)
               {
                  Changelog cObj =
                  (Changelog)viewer.getElementAt(c);
                  changelogs.add(cObj);
               }

           //create Changelogstore, assigning Changelog
            ChangelogStore changelogstore = new ChangelogStore();
            changelogstore.setName(filename+" Changelog");
            changelogstore.setChangelogList(changelogs);

            // create JAXB context and instantiate marshaller
            try
            {
               JAXBContext context =
               JAXBContext.newInstance(ChangelogStore.class);
               Marshaller m = context.createMarshaller();
               m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
                  Boolean.TRUE);

               Writer w = null;
```

```
                              try {
                                    w = new FileWriter(CHANGELOG_XML);
                                    m.marshal(changelogstore, w);
                              } finally {
                                    try {
                                          w.close();
                                    } catch (Exception e) {}
                              }
                        }catch(Exception e)
                        {
                              e.printStackTrace();

                              if(e.getClass().toString().equals("class
java.io.FileNotFoundException"))
                              {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                              "Unable to save Changelog as xml. \nSystem
cannot find the PATH specified. \nPlease change the settings appropriately",
"Changelog Warning",JOptionPane.WARNING_MESSAGE);
                              }
                        }
                  }
                  }
            };
            saveas.setText("Save As");
            saveas.setToolTipText("SaveAs Changelog");

      saveas.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_ETOOL_SAVEALL_EDIT));



            open = new Action() {

                  public void run()
                  {
                        String directoryFile = new
                              FileBrowse().getDirectoryFile();
                        CHANGELOGOPEN_XML = directoryFile;

                        try
                        {
                              if(!CHANGELOGOPEN_XML.equals("") &&
CHANGELOGOPEN_XML.endsWith("xml") || CHANGELOGOPEN_XML.endsWith("XML"))
                              {

                        //Remove current elements before inserting
                        int tableCount = viewer.getTable().getItemCount();
                              viewer.getTable().remove(0,tableCount-1);

                        try
                        {
                              JAXBContext context =
                        JAXBContext.newInstance(ChangelogStore.class);
                        Unmarshaller um = context.createUnmarshaller();
```

```
                ChangelogStore changelogstore2 = (ChangelogStore)
                um.unmarshal(new FileReader(
                CHANGELOGOPEN_XML));


                //Insert elements into Changelog
                if(changelogstore2 instanceof ChangelogStore)
                {
                for (int i = 0; i <
                changelogstore2.getChangelogsList().toArray().length;
                i++)
                {
                    Changelog cObj =
                    (Changelog)(changelogstore2.getChangelogsList()
                    .get(i));
                    viewer.insert(cObj,i);
                }
                }
                }
                catch(UnmarshalException ex)
                {
                    JOptionPane.showMessageDialog(null,
                    "Content is inappropriate. Please select the
correct file","Changelog Warning",JOptionPane.WARNING_MESSAGE);
                }

            }
            else
            {
                //Java Swing dialog box
                JOptionPane.showMessageDialog(null,
                "Content is inappropriate. Please select the correct
                file","Changelog Warning",
                JOptionPane.WARNING_MESSAGE);
            }

            }
            catch(Exception e)
            {
                e.printStackTrace();
            }
            }
        };
        open.setText("Open");
        open.setToolTipText("Open Changelog");

    open.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
            getImageDescriptor(ISharedImages.IMG_OBJ_FOLDER));

        insert = new Action() {
            public void run() {
            //Insert row logic goes here
            IStructuredSelection selection =
                (IStructuredSelection)viewer.getSelection();

            if ( (viewer.getTable().getItemCount() >= 1) &&
                (selection.isEmpty()) )
```

```
                        {
                              //Java Swing dialog box
                        JOptionPane.showMessageDialog(null,
                        "Please select a row before you INSERT a NEW ROW
above it","Changelog Warning",JOptionPane.WARNING_MESSAGE);
                        }
                        else if(viewer.getTable().getItemCount() == 0)
                        {
                              Changelog cgObj = new Changelog();
                              viewer.insert(cgObj, 0);

                        }else if(!(selection.isEmpty()) &&
                              (viewer.getTable().getItemCount() >= 1))
                        {

                              for(Object o:selection.toList()){

                                 Changelog cgObj = new Changelog();
                                 o = cgObj;
                                 if( o instanceof Changelog)
                                 {
                                    viewer.insert(o,
                                    viewer.getTable().getSelectionIndex());
                                 }
                              }
                        }

                  }
            };
            insert.setText("Insert");
            insert.setToolTipText("Insert new row");

      insert.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                  getImageDescriptor(ISharedImages.IMG_OBJ_ADD));



            delete = new Action() {
                  public void run() {
                        //Delete row logic goes here
                        IStructuredSelection selection =
(IStructuredSelection)viewer.getSelection();

                        if ( (viewer.getTable().getItemCount() >= 1) &&
                              (selection.isEmpty()) )
                        {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                                  "Please select a row before you DELETE it",
                                  "Changelog Warning",
                                  JOptionPane.WARNING_MESSAGE);
                        }else if(viewer.getTable().getItemCount() == 0)
                        {
                              //Java Swing dialog box
                              JOptionPane.showMessageDialog(null,
                                  "There are no rows to DELETE",
                                  "Changelog Warning",
                                  JOptionPane.WARNING_MESSAGE);

                        }else if(!(selection.isEmpty()))
                        {
                              int index =
                              viewer.getTable().getSelectionIndex();
                              viewer.getTable().remove(index);
                        }
                  }
            };
            delete.setText("Delete");
            delete.setToolTipText("Delete row");

      delete.setImageDescriptor(PlatformUI.getWorkbench().getSharedImages().
                        getImageDescriptor(ISharedImages.IMG_ETOOL_DELETE));

      }



      /**
       * Passing the focus request to the viewer's control.
       */
      public void setFocus() {
            viewer.getControl().setFocus();
      }

      @Override
      public void selectionChanged(IWorkbenchPart part, ISelection selection)
      {

      }


      //Function to load the properties file
      private static Properties loadProperties(String fileName)
      {
            InputStream propsFile;
        Properties tempProp = new Properties();

        try
        {
            propsFile = new FileInputStream(fileName);
            tempProp.load(propsFile);
            propsFile.close();
        }catch (IOException ioe) {
            System.out.println("I/O Exception.");
            ioe.printStackTrace();
            System.exit(0);
        }

        return tempProp;

    }

}
```

CellEditingChangelog.java

```
/*
 * CellEditingChangelog to support editing the Changelog rows
 */
package org.example.tableviewer.views;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;

import org.eclipse.jface.viewers.CellEditor;
import org.eclipse.jface.viewers.ColumnViewer;
import org.eclipse.jface.viewers.ComboBoxCellEditor;
import org.eclipse.jface.viewers.EditingSupport;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.jface.viewers.TextCellEditor;

public class CellEditingChangelog extends EditingSupport {
      private CellEditor editor;
      private int column;

      final String PROPFILE=
"C:/Users/pzs0011/workspace/org.example.TableViewer/attributes.properties";
      Properties myProp;

      public CellEditingChangelog(ColumnViewer viewer, int column) {
            super(viewer);

            //Load the property file before use
            myProp = loadProperties(PROPFILE);

            //Please Note: The phases/activities are read from PROPERTY FILE
            (Key=Value) pair.
            String[] type = new String[11];
            type[0] = myProp.getProperty("defect_type_1");
            type[1] = myProp.getProperty("defect_type_2");
            type[2] = myProp.getProperty("defect_type_3");
            type[3] = myProp.getProperty("defect_type_4");
            type[4] = myProp.getProperty("defect_type_5");
            type[5] = myProp.getProperty("defect_type_6");
            type[6] = myProp.getProperty("defect_type_7");
            type[7] = myProp.getProperty("defect_type_8");
            type[8] = myProp.getProperty("defect_type_9");
            type[9] = myProp.getProperty("defect_type_10");
            type[10] = myProp.getProperty("defect_type_11");

            //Please Note: The phases/activities are read from PROPERTY FILE
            (Key=Value) pair.
            String[] activity = new String[10];
            activity[0] = myProp.getProperty("activity_1");
```

```
activity[1] = myProp.getProperty("activity_2");
activity[2] = myProp.getProperty("activity_3");
activity[3] = myProp.getProperty("activity_4");
activity[4] = myProp.getProperty("activity_5");
activity[5] = myProp.getProperty("activity_6");
activity[6] = myProp.getProperty("activity_7");
activity[7] = myProp.getProperty("activity_8");
activity[8] = myProp.getProperty("activity_9");
activity[9] = myProp.getProperty("activity_10");


// Create the correct editor based on the column index
switch (column) {

case 0:
      editor = new TextCellEditor(((TableViewer)
                                    viewer).getTable());
      break;
case 1:
      editor = new TextCellEditor(((TableViewer)
                                    viewer).getTable());
      break;
case 2:
      editor = new ComboBoxCellEditor(((TableViewer)
                              viewer).getTable(),type);
      break;
case 3:
      editor = new ComboBoxCellEditor(((TableViewer)
                              viewer).getTable(),activity);
      break;
case 4:
      editor = new TextCellEditor(((TableViewer)
                                          viewer).getTable());
      break;
case 5:
      editor = new ComboBoxCellEditor(((TableViewer)
                                    viewer).getTable(),activity);
      break;
case 6:
      editor = new TextCellEditor(((TableViewer)
                                          viewer).getTable());
      break;
case 7:
      editor = new TextCellEditor(((TableViewer)
                                          viewer).getTable());
      break;
case 8:
      editor = new TextCellEditor(((TableViewer)
                                          viewer).getTable());
      break;
case 9:
      editor = new TextCellEditor(((TableViewer)
                                          viewer).getTable());
      break;
default:
      editor = new TextCellEditor(((TableViewer)
                                          viewer).getTable());
      }
      this.column = column;

}

@Override
protected boolean canEdit(Object element) {
      return true;
}

@Override
protected CellEditor getCellEditor(Object element) {
      return editor;
}

@Override
protected Object getValue(Object element) {
      Changelog cObj = (Changelog) element;

      switch (this.column) {
      case 0:
            return cObj.getNumber();
      case 1:
            return cObj.getDate();
      case 2:
            //Please Note: The defect types are read from PROPERTY FILE
            (Key=Value) pair.
            if (cObj.getType().equals(
                  myProp.getProperty("defect_type_1") )) {
                  return 0;
            }
            else if (cObj.getType().equals(
                  myProp.getProperty("defect_type_2") )) {
                  return 1;
            }
            else if(cObj.getType().equals(
                  myProp.getProperty("defect_type_3") )) {
                  return 2;
            }
            else if(cObj.getType().equals(
                  myProp.getProperty("defect_type_4") )) {
                  return 3;
            }
            else if(cObj.getType().equals(
                  myProp.getProperty("defect_type_5") )) {
                  return 4;
            }
            else if(cObj.getType().equals(
                  myProp.getProperty("defect_type_6") )) {
                  return 5;
            }
            else if(cObj.getType().equals(
                  myProp.getProperty("defect_type_7") )) {
                  return 6;
            }
            else if(cObj.getType().equals(
                  myProp.getProperty("defect_type_8") )) {
```

```
                return 7;
            }
            else if(cObj.getType().equals(
                myProp.getProperty("defect_type_9") )) {
                return 8;
            }
            else if(cObj.getType().equals(
                myProp.getProperty("defect_type_10") )) {
                return 9;
            }
            else if(cObj.getType().equals(
                myProp.getProperty("defect_type_11") )) {
                return 10;
            }

        case 3:
            //Please Note: The phases/activities are read from PROPERTY
            FILE (Key=Value) pair.
            if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_1") )) {
                return 0;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_2") )) {
                return 1;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_3") )) {
                return 2;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_4") )) {
                return 3;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_5") )) {
                return 4;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_6") )) {
                return 5;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_7") )) {
                return 6;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_8") )) {
                return 7;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_9") )) {
                return 8;
            }
            else if (cObj.getInjectactivity().equals(
                myProp.getProperty("activity_10") )) {
                return 9;
```

```
      }

case 4:
      return cObj.getInjectfeatureset();
case 5:
      //Please Note: The phases/activities are read from PROPERTY
      FILE (Key=Value) pair.
      if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_1") )) {
            return 0;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_2") )) {
            return 1;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_3") )) {
            return 2;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_4") )) {
            return 3;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_5") )) {
            return 4;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_6") )) {
            return 5;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_7") )) {
            return 6;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_8") )) {
            return 7;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_9") )) {
            return 8;
      }
      else if (cObj.getRemoveactivity().equals(
            myProp.getProperty("activity_10") )) {
            return 9;
      }

case 6:
      return cObj.getRemovefeatureset();
case 7:
      return cObj.getFixtime();
case 8:
      return cObj.getFixreference();
case 9:
      return cObj.getDescription();
default:
        break;
    }
    return null;
}

@Override
protected void setValue(Object element, Object value) {
    Changelog cObj = (Changelog) element;

    switch (this.column) {
    case 0:
        cObj.setNumber(String.valueOf(value));
        break;
    case 1:
        cObj.setDate(String.valueOf(value));
        break;
    case 2:
        //Please Note: The defect types are read from PROPERTY FILE
        (Key=Value) pair.
        if (((Integer) value) == 0) {
            cObj.setType( myProp.getProperty("defect_type_1") );
        } else if (((Integer) value) == 1) {
            cObj.setType( myProp.getProperty("defect_type_2") );
        } else if(((Integer) value) == 2) {
            cObj.setType( myProp.getProperty("defect_type_3") );
        } else if(((Integer) value) == 3) {
            cObj.setType( myProp.getProperty("defect_type_4") );
        } else if(((Integer) value) == 4) {
            cObj.setType( myProp.getProperty("defect_type_5") );
        } else if(((Integer) value) == 5) {
            cObj.setType( myProp.getProperty("defect_type_6") );
        } else if(((Integer) value) == 6) {
            cObj.setType( myProp.getProperty("defect_type_7") );
        } else if(((Integer) value) == 7) {
            cObj.setType( myProp.getProperty("defect_type_8") );
        } else if(((Integer) value) == 8) {
            cObj.setType( myProp.getProperty("defect_type_9") );
        } else if(((Integer) value) == 9) {
            cObj.setType( myProp.getProperty("defect_type_10") );
        } else if(((Integer) value) == 10) {
            cObj.setType( myProp.getProperty("defect_type_11") );
        }
        break;
    case 3:
        //Please Note: The phases/activities are read from PROPERTY
        FILE (Key=Value) pair.
        if (((Integer) value) == 0) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_1") );
        } else if (((Integer) value) == 1) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_2") );
        } else if (((Integer) value) == 2) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_3") );
        } else if(((Integer) value) == 3) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_4") );
      } else if(((Integer) value) == 4) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_5") );
      } else if(((Integer) value) == 5) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_6") );
      } else if(((Integer) value) == 6) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_7") );
      } else if(((Integer) value) == 7) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_8") );
      } else if(((Integer) value) == 8) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_9") );
      } else if(((Integer) value) == 9) {
            cObj.setInjectactivity(
            myProp.getProperty("activity_10") );
      }
      break;
case 4:
      cObj.setInjectfeatureset(String.valueOf(value));
      break;
case 5:
      //Please Note: The phases/activities are read from PROPERTY
      FILE (Key=Value) pair.
      if (((Integer) value) == 0) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_1") );
      } else if (((Integer) value) == 1) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_2") );
      } else if (((Integer) value) == 2) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_3") );
      } else if(((Integer) value) == 3) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_4") );
      } else if(((Integer) value) == 4) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_5") );
      } else if(((Integer) value) == 5) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_6") );
      } else if(((Integer) value) == 6) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_7") );
      } else if(((Integer) value) == 7) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_8") );
      } else if(((Integer) value) == 8) {
            cObj.setRemoveactivity(
            myProp.getProperty("activity_9") );
      } else if(((Integer) value) == 9) {
```

```
                        cObj.setRemoveactivity(
                        myProp.getProperty("activity_10") );
                  }
                  break;
            case 6:
                  cObj.setRemovefeatureset(String.valueOf(value));
                  break;
            case 7:
                  cObj.setFixtime(String.valueOf(value));
                  break;
            case 8:
                  cObj.setFixreference(String.valueOf(value));
                  break;
            case 9:
                  cObj.setDescription(String.valueOf(value));
                  break;
            default:
                  break;
            }

            getViewer().update(element, null);
      }

      //Function to load the properties file
      private static Properties loadProperties(String fileName)
      {
            InputStream propsFile;
        Properties tempProp = new Properties();

        try
        {
            propsFile = new FileInputStream(fileName);
            tempProp.load(propsFile);
            propsFile.close();
        }catch (IOException ioe) {
            System.out.println("I/O Exception.");
            ioe.printStackTrace();
            System.exit(0);
        }

        return tempProp;

    }

}
```

SizeMatrix.java

```
/* SizeMatrix class to represent the SizeMatrix data members
 * with getter and setter methods
 */
package org.example.tableviewer.views;

public class SizeMatrix {
      private String strBucket = "";
      private String strLow = "";
      private String strMid = "";
      private String strHigh = "";

      public SizeMatrix()
      {
            this.strBucket = "";
            this.strLow = "";
            this.strMid = "";
            this.strHigh = "";

      }

      public String getStrBucket() {
            return strBucket;
      }

      public void setStrBucket(String strBucket) {
            this.strBucket = strBucket;
      }

      public String getStrLow() {
            return strLow;
      }

      public void setStrLow(String strLow) {
            this.strLow = strLow;
      }

      public String getStrMid() {
            return strMid;
      }

      public void setStrMid(String strMid) {
            this.strMid = strMid;
      }

      public String getStrHigh() {
            return strHigh;
      }

      public void setStrHigh(String strHigh) {
            this.strHigh = strHigh;
      }


}
```

PartInformation.java

```
/* PartInformation class to represent the parts information data members
```

```
 * with getter and setter methods
 */
package org.example.tableviewer.views;

public class PartInformation {

      private String strProjects = "";
      private String strProxyName = "";
      private String strLOC = "";
      private String strNoOfMethods = "";
      private String strRelSize = "";

      public PartInformation()
      {
            this.strProjects = "";
            this.strProxyName = "";
            this.strLOC = "";
            this.strNoOfMethods = "";
            this.strRelSize = "";
      }

      public String getStrProjects() {
            return strProjects;
      }

      public void setStrProjects(String strProjects) {
            this.strProjects = strProjects;
      }

      public String getStrProxyName() {
            return strProxyName;
      }

      public void setStrProxyName(String strProxyName) {
            this.strProxyName = strProxyName;
      }

      public String getStrLOC() {
            return strLOC;
      }

      public void setStrLOC(String strLOC) {
            this.strLOC = strLOC;
      }

      public String getStrNoOfMethods() {
            return strNoOfMethods;
      }

      public void setStrNoOfMethods(String strNoOfMethods) {
            this.strNoOfMethods = strNoOfMethods;
      }

      public String getStrRelSize() {
            return strRelSize;
      }
```

```
    public void setStrRelSize(String strRelSize) {
        this.strRelSize = strRelSize;
    }

}
```

SizeMatrixView.java

```
/*
 * SizeMatrixView demonstrates the SizeMatrix plug-in view
 */
package org.example.tableviewer.views;

import org.eclipse.jface.viewers.ArrayContentProvider;
import org.eclipse.jface.viewers.ISelection;
import org.eclipse.jface.viewers.IStructuredContentProvider;
import org.eclipse.jface.viewers.IStructuredSelection;
import org.eclipse.jface.viewers.ITableLabelProvider;
import org.eclipse.jface.viewers.LabelProvider;
import org.eclipse.jface.viewers.TableViewer;
import org.eclipse.jface.viewers.TableViewerColumn;
import org.eclipse.jface.viewers.Viewer;
import org.eclipse.swt.SWT;
import org.eclipse.swt.graphics.Image;
import org.eclipse.swt.layout.FillLayout;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.layout.RowLayout;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Table;
import org.eclipse.ui.ISelectionListener;
import org.eclipse.ui.ISharedImages;
import org.eclipse.ui.IWorkbenchPart;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.forms.widgets.Form;
import org.eclipse.ui.forms.widgets.FormToolkit;
import org.eclipse.ui.part.ViewPart;


public class SizeMatrixView extends ViewPart implements ISelectionListener {
public static final String ID = "org.example.tableviewer.views.SampleView";

    public static TableViewer smViewer;
    public static TableViewer piViewer;

    private Form form;

    private String[] smColumnNames;
    private String[] piColumnNames;

    public static int piviewerRowIndexCnt = 0;
    public static int smviewerRowIndexCnt = 0;
```

```
PartInformation piObj = new PartInformation();


class ViewContentProvider implements IStructuredContentProvider {
      public void inputChanged(Viewer v, Object oldInput, Object
      newInput) {
      }

      public void dispose() {
      }

      public Object[] getElements(Object parent) {
            return new String[] { "One", "Two", "Three" };

      }
}



class ViewLabelProvider extends LabelProvider implements
            ITableLabelProvider {
      public String getColumnText(Object obj, int index) {
            return getText(obj);
      }

      public Image getColumnImage(Object obj, int index) {
            return getImage(obj);
      }

      public Image getImage(Object obj) {
            return
      PlatformUI.getWorkbench().getSharedImages().getImage(
                        ISharedImages.IMG_OBJ_ELEMENT);
      }
}


/**
 * This is a callback that will allow us to create the viewer and
      initialize it.
 */
public void createPartControl(Composite parent) {
      FormToolkit toolkit = new FormToolkit(parent.getDisplay());

      getSite().getPage().addSelectionListener(this);

      toolkit = new FormToolkit(parent.getDisplay());
      form = toolkit.createForm(parent);
      form.setText("Size Matrix Calculation:");
      toolkit.decorateFormHeading(form);

      GridLayout layout = new GridLayout();
  form.getBody().setLayout(layout);


  Group sizeMatrixGroup = new Group(form.getBody(), SWT.SHADOW_IN);
  sizeMatrixGroup.setText("Size Matrix Calculation from Version Control
      (SVN):");

      RowLayout rowLayout = new RowLayout();
      rowLayout.wrap = false;
      rowLayout.pack = false;
      rowLayout.justify = true;
      rowLayout.type = SWT.HORIZONTAL;
      rowLayout.marginLeft = 5;
      rowLayout.marginTop = 5;
      rowLayout.marginRight = 5;
      rowLayout.marginBottom = 5;
      rowLayout.spacing = 25;
      sizeMatrixGroup.setLayout(rowLayout);

      //Parts Information Group
  Group piGroup = new Group(sizeMatrixGroup, SWT.SHADOW_IN);
      piGroup.setText("Parts Information:");
  piGroup.setLayout(new GridLayout(1, false));


piViewer = new TableViewer(piGroup, SWT.MULTI | SWT.H_SCROLL |
                                    SWT.V_SCROLL | SWT.FULL_SELECTION);
final Table pitable = piViewer.getTable();

      pitable.setHeaderVisible(true);
      pitable.setLinesVisible(true);

      piColumnNames = new String[] {"Projects", "Proxy Name", "LOC",
                                    "No. Of Methods", "Relative Size"};
      int[] piColumnWidths = new int[] {125,125,100,100,100};
      int[] piColumnAlignments = new int[] {

SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER};


      for(int j=0; j < piColumnNames.length;j++)
      {

            TableViewerColumn piViewerColumn = new
                        TableViewerColumn(piViewer, SWT.NONE);

            piViewerColumn.getColumn().setText(piColumnNames[j]);
            piViewerColumn.getColumn().setWidth(piColumnWidths[j]);

piViewerColumn.getColumn().setAlignment(piColumnAlignments[j]);
            piViewerColumn.getColumn().setResizable(true);
            piViewerColumn.getColumn().setMoveable(true);
      }

piViewer.setLabelProvider(new PartInformationTableLabelProvider());
piViewer.setContentProvider(new ArrayContentProvider());

      PartInformation piObj = new PartInformation();


      piViewer.add(piObj);
      piViewer.add(piObj);
      piViewer.add(piObj);
      piViewer.add(piObj);
      piViewer.add(piObj);
      piViewer.add(piObj);


       //Size Matrix Group
  Group smGroup = new Group(sizeMatrixGroup, SWT.SHADOW_IN);
  smGroup.setText("Size Matrix:");
  smGroup.setLayout(new FillLayout());


      smViewer = new TableViewer(smGroup, SWT.MULTI | SWT.H_SCROLL |
                              SWT.V_SCROLL | SWT.FULL_SELECTION);
  final Table smtable = smViewer.getTable();

      smtable.setHeaderVisible(true);
      smtable.setLinesVisible(true);

      smColumnNames = new String[] {"","Low", "Mid", "High"};
      int[] smColumnWidths = new int[] {75,75,75,75};
      int[] smColumnAlignments = new int[] {
                  SWT.CENTER,SWT.CENTER,SWT.CENTER,SWT.CENTER};

      for(int i=0; i<smColumnNames.length;i++)
      {
            TableViewerColumn smViewerColumn = new
                        TableViewerColumn(smViewer, SWT.NONE);

            smViewerColumn.getColumn().setText(smColumnNames[i]);
            smViewerColumn.getColumn().setWidth(smColumnWidths[i]);

smViewerColumn.getColumn().setAlignment(smColumnAlignments[i]);
            smViewerColumn.getColumn().setResizable(true);
            smViewerColumn.getColumn().setMoveable(true);
      }

      smViewer.setLabelProvider(new SizeMatrixTableLabelProvider());
      smViewer.setContentProvider(new ArrayContentProvider());


      // Create the help context id for the viewer's control

PlatformUI.getWorkbench().getHelpSystem().setHelp(smViewer.getControl()
                              , "org.example.TableViewer.viewer");
PlatformUI.getWorkbench().getHelpSystem().setHelp(piViewer.getControl()
                              , "org.example.TableViewer.viewer");
      makeActions();
      hookContextMenu();
      hookDoubleClickAction();
      contributeToActionBars();


}
```

```
public class SizeMatrixTableLabelProvider extends LabelProvider
                                    implements ITableLabelProvider
{
      public Image getColumnImage( Object element, int index)
      {
            return null;

      }


      public String getColumnText(Object element, int index)
      {

            SizeMatrix smObj = (SizeMatrix) element;
            switch(index){
            case 0:
                  return smObj.getStrBucket();
            case 1:
                  return smObj.getStrLow();
            case 2:
                  return smObj.getStrMid();
            case 3:
                  return smObj.getStrHigh();
            default:
                  return "unknown" + index;
            }

      }

}


public class PartInformationTableLabelProvider extends LabelProvider
                                    implements ITableLabelProvider
{
      public Image getColumnImage( Object element, int index)
      {
            return null;

      }

      public String getColumnText(Object element, int index)
      {
            PartInformation piObj = (PartInformation) element;
            switch(index){
            case 0:
                  return piObj.getStrProjects();
            case 1:
                  return piObj.getStrProxyName();
            case 2:
                  return piObj.getStrLOC();
            case 3:
                  return piObj.getStrNoOfMethods();
            case 4:
                  return piObj.getStrRelSize();
            default:
                  return "unknown" + index;
            }
      }

}



private void contributeToActionBars() {
      // TODO Auto-generated method stub
}

private void hookDoubleClickAction() {
      // TODO Auto-generated method stub
}

private void hookContextMenu() {
      // TODO Auto-generated method stub
}

private void makeActions() {

}

/**
 * Passing the focus request to the viewer's control.
 */
public void setFocus() {
      smViewer.getControl().setFocus();
      piViewer.getControl().setFocus();
}


@Override
public void selectionChanged(IWorkbenchPart part, ISelection selection)
{

      if(!(selection instanceof IStructuredSelection))
            return;
      IStructuredSelection ss = (IStructuredSelection) selection;

      Object o = ss.getFirstElement();

      if( o instanceof PartInformation)
      {
            piViewer.insert(o, piviewerRowIndexCnt);
            piviewerRowIndexCnt = piviewerRowIndexCnt + 1;

      }
      else if (o instanceof SizeMatrix)
      {
            smViewer.insert(o, smviewerRowIndexCnt);
            smviewerRowIndexCnt = smviewerRowIndexCnt + 1;

      }
}
```

```
}
```

ProjectDashboardView.java

```
/*
 * ProjectDashboardView demonstrates the Process Dashboard plug-in view
 */
package org.example.tableviewer.views;


import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.text.SimpleDateFormat;

import java.util.Calendar;
import java.util.Properties;

import javax.swing.JOptionPane;

import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Group;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Spinner;
import org.eclipse.swt.widgets.Text;
import org.eclipse.ui.forms.widgets.Form;
import org.eclipse.ui.forms.widgets.FormToolkit;
import org.eclipse.ui.part.*;
import org.eclipse.core.runtime.ListenerList;
import org.eclipse.jface.viewers.*;
import org.eclipse.swt.graphics.Image;
import org.eclipse.swt.layout.FillLayout;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.layout.RowLayout;
import org.eclipse.jface.action.*;
import org.eclipse.jface.dialogs.MessageDialog;
import org.eclipse.ui.*;
import org.eclipse.swt.widgets.Menu;
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.*;



/**
 * This class demonstrates how to plug-in a new
 * workbench view.The view is connected to the model using
 * a content provider.The view uses a label provider to define
 * how model objects should be presented in the view. Each
 * view can present the same model objects using
 * different labels and icons, if needed. Alternatively,
 * a single label provider can be shared between views
 * in order to ensure that objects of the same type are
 * presented in the same way everywhere.
 *
 */

public class ProjectDashboardView extends ViewPart implements
ISelectionProvider{

      /**
       * The ID of the view as specified by the extension.
       */
      public static final String ID = "projectdashboard.views.SampleView";

      final String PROPFILE=
"C:/Users/pzs0011/workspace/org.example.TableViewer/attributes.properties";
      Properties myProp;

      private TableViewer viewer;
      private Action action1;
      private Action action2;
      private Action doubleClickAction;

      Timelog globalTlg = new Timelog();
      Timelog newglobalTlg = new Timelog();

      private Group flowgraphGroup;

      private Button activity1But;
      private Button activity2But;
      private Button activity3But;
      private Button activity4But;
      private Button activity5But;
      private Button activity6But;
      private Button activity7But;
      private Button activity8But;
      private Button activity9But;
      private Button activity10But;


      private int redrawflag = 0;

      private FormToolkit toolkit;
      private Form form;

      ListenerList listeners = new ListenerList();

      static int intStartButPressFlag = 0;
      static int intPhaseAfterStartFlag = 0;

      public static boolean testflag = false;


      /*
       * The content provider class is responsible for
       * providing objects to the view. It can wrap
       * existing objects in adapters or simply return
       * objects as-is. These objects may be sensitive
       * to the current input of the view, or ignore
       * it and always show the same content
       * (like Task List, for example).
       */

      class ViewContentProvider implements IStructuredContentProvider {
            public void inputChanged(Viewer v, Object oldInput, Object
            newInput) {
            }
            public void dispose() {
            }
            public Object[] getElements(Object parent) {
                  return new String[] {"one","two","three"};
            }

      }
      class ViewLabelProvider extends LabelProvider implements
            ITableLabelProvider {
            public String getColumnText(Object obj, int index) {
                  return getText(obj);
            }
            public Image getColumnImage(Object obj, int index) {
                  return getImage(obj);
            }
            public Image getImage(Object obj) {
                  return PlatformUI.getWorkbench().

      getSharedImages().getImage(ISharedImages.IMG_OBJ_ELEMENT);
            }
      }
      class NameSorter extends ViewerSorter {
      }

      //Constructor
      public ProjectDashboardView() {
      }

      /**
       * This is a callback that will allow us
       * to create the viewer and initialize it.
       */
      public void createPartControl(Composite parent) {

            getSite().setSelectionProvider(this);
            toolkit = new FormToolkit(parent.getDisplay());
            form = toolkit.createForm(parent);
            form.setText("Process Dashboard");
            toolkit.decorateFormHeading(form);

        GridLayout layout = new GridLayout();
        form.getBody().setLayout(layout);

        //Phases Group
        Group phGroup = new Group(form.getBody(), SWT.SHADOW_IN);
        phGroup.setText("Activities:");
        phGroup.setLayout(new FillLayout());
```

```
//Load the property file before use
myProp = loadProperties(PROPFILE);

activity1But = new Button(phGroup, SWT.TOGGLE);
activity1But.setText(" "+myProp.getProperty("activity_1_symbol"));
activity1But.setSelection(false);

activity2But = new Button(phGroup, SWT.TOGGLE);
activity2But.setText(myProp.getProperty("activity_2_symbol"));
activity2But.setSelection(false);

activity3But = new Button(phGroup, SWT.TOGGLE);
activity3But.setText(myProp.getProperty("activity_3_symbol"));
activity3But.setSelection(false);

activity4But = new Button(phGroup, SWT.TOGGLE);
activity4But.setText(myProp.getProperty("activity_4_symbol"));
activity4But.setSelection(false);

activity5But = new Button(phGroup, SWT.TOGGLE);
activity5But.setText(myProp.getProperty("activity_5_symbol"));
activity5But.setSelection(false);

activity6But = new Button(phGroup, SWT.TOGGLE);
activity6But.setText(myProp.getProperty("activity_6_symbol"));
activity6But.setSelection(false);

activity7But = new Button(phGroup, SWT.TOGGLE);
activity7But.setText(myProp.getProperty("activity_7_symbol"));
activity7But.setSelection(false);

activity8But = new Button(phGroup, SWT.TOGGLE);
activity8But.setText(myProp.getProperty("activity_8_symbol"));
activity8But.setSelection(false);

activity9But = new Button(phGroup, SWT.TOGGLE);
activity9But.setText(myProp.getProperty("activity_9_symbol"));
activity9But.setSelection(false);


//Comments Group
Group comGroup = new Group(form.getBody(), SWT.SHADOW_IN);
comGroup.setText("Enter Comments:");
comGroup.setLayout(new GridLayout(1, false));

final Text text = new Text(comGroup, SWT.BORDER | SWT.WRAP |
                                                    SWT.MULTI);
    GridData gridData = new GridData();
    gridData.horizontalAlignment = SWT.FILL;
    gridData.grabExcessHorizontalSpace = true;
    gridData.verticalAlignment = SWT.FILL;
    gridData.grabExcessVerticalSpace = true;
    text.setLayoutData(gridData);

    text.setText("                                  \n\n");
```

```
      Group sptimGroup = new Group(form.getBody(), SWT.SHADOW_IN);
        sptimGroup.setText("");

      RowLayout rowLayout = new RowLayout();
      rowLayout.wrap = false;
      rowLayout.pack = false;
      rowLayout.justify = true;
      rowLayout.type = SWT.HORIZONTAL;
      rowLayout.marginLeft = 5;
      rowLayout.marginTop = 5;
      rowLayout.marginRight = 5;
      rowLayout.marginBottom = 5;
      rowLayout.spacing = 40;
      sptimGroup.setLayout(rowLayout);

      //FeatureSet Group
      Group fsGroup = new Group(sptimGroup, SWT.SHADOW_IN);
      fsGroup.setText("Feature Set:");
      fsGroup.setLayout(new GridLayout(2, false));

            Label label = new Label(fsGroup, SWT.NULL);
            label.setText("      ");


        final Spinner spinner = new Spinner (fsGroup, SWT.BORDER |
            SWT.CENTER);
            spinner.setMinimum(0);
            spinner.setMaximum(100);
            spinner.setSelection(0);
            spinner.setIncrement(1);
            spinner.setPageIncrement(100);
            spinner.pack();


      //Timer Group
      Group timGroup = new Group(sptimGroup, SWT.SHADOW_IN);
  timGroup.setText("Timer:");
  timGroup.setLayout(new GridLayout(1, true));

  final Button startstopBut = new Button(timGroup, SWT.TOGGLE);
  startstopBut.setText("        START        ");

//Code Complete Group
  Group codecompGroup = new Group(form.getBody(), SWT.SHADOW_IN);
  codecompGroup.setText("Code Complete:");
  codecompGroup.setLayout(new GridLayout(1, true));

  activity10But = new Button(codecompGroup, SWT.PUSH);

  activity10But.setText("
         "+myProp.getProperty("activity_10").toUpperCase()+"
      ");


  //Flow Graph Group
  flowgraphGroup = new Group(form.getBody(), SWT.SHADOW_IN );
  flowgraphGroup.setText("Flow Graph:");
```

```
        FillLayout fillLayout = new FillLayout();
        fillLayout.type = SWT.VERTICAL;
        fillLayout.marginHeight = 135;
        fillLayout.marginWidth = 136;
        flowgraphGroup.setLayout(fillLayout);
        flowgraphGroup.addPaintListener(new FlowGraphPaintListener());


        //Planning Overall action(button pressed) detected here
        activity1But.addMouseListener(new MouseAdapter() {

          public void mouseDown(MouseEvent e) {

            System.out.println("Planning Overall Button Pressed");
                        //planOvBut.setSelection(true);
                        activity2But.setSelection(false);
                        activity3But.setSelection(false);
                        activity4But.setSelection(false);
                        activity5But.setSelection(false);
                        activity6But.setSelection(false);
                        activity7But.setSelection(false);
                        activity8But.setSelection(false);
                        activity9But.setSelection(false);


                        if(!activity1But.getSelection())
                        {
                              redrawflag = 1;
                        }
                        else if(activity1But.getSelection())
                        {
                              redrawflag = 0;
                        }

                        flowgraphGroup.redraw();


                  //Please Note:Checking for NOT condition because the
    planOvBut button is RESET in one part of the code before we reach this point
                        if(!activity1But.getSelection())
                        {

      phaseAfterStart(myProp.getProperty("activity_1"),text,spinner);
                        }
                  }

            });


        //Planning Iterative action(button pressed) detected here
        activity2But.addMouseListener(new MouseAdapter() {

          public void mouseDown(MouseEvent e) {


            System.out.println("Planning Iterative Button Pressed");
                        activity1But.setSelection(false);
                        //planItBut.setSelection(true);
                        activity3But.setSelection(false);
                        activity4But.setSelection(false);
                        activity5But.setSelection(false);
                        activity6But.setSelection(false);
                        activity7But.setSelection(false);
                        activity8But.setSelection(false);
                        activity9But.setSelection(false);


                        if(!activity2But.getSelection())
                        {
                              redrawflag = 2;

                        }
                        else if(activity2But.getSelection())
                        {
                              redrawflag = 0;
                        }

                        flowgraphGroup.redraw();

                        //Please Note:Checking for NOT condition because the
planItBut button is RESET in one part of the code before we reach this point
                        if(!activity2But.getSelection())
                        {

      phaseAfterStart(myProp.getProperty("activity_2"),text,spinner);
                        }

          }
            });


        //Architecture action(button pressed) detected here
        activity3But.addMouseListener(new MouseAdapter() {

          public void mouseDown(MouseEvent e) {
                        System.out.println("Architecture Button Pressed");
                        activity1But.setSelection(false);
                        activity2But.setSelection(false);
                        activity4But.setSelection(false);
                        activity5But.setSelection(false);
                        activity6But.setSelection(false);
                        activity7But.setSelection(false);
                        activity8But.setSelection(false);
                        activity9But.setSelection(false);

                        if(!activity3But.getSelection())
                        {
                              redrawflag = 3;

                        }
                        else if(activity3But.getSelection())
                        {
                              redrawflag = 0;
                        }
            flowgraphGroup.redraw();


            //Please Note:Checking for NOT condition because the
archiBut button is RESET in one part of the code before we reach this point
            if(!activity3But.getSelection())
            {

   phaseAfterStart(myProp.getProperty("activity_3"),text,spinner);
            }
         }

      });


    //Construction action(button pressed) detected here
    activity4But.addMouseListener(new MouseAdapter() {

     public void mouseDown(MouseEvent e) {
            System.out.println("Construction Button Pressed");
            activity1But.setSelection(false);
            activity2But.setSelection(false);
            activity3But.setSelection(false);
            activity5But.setSelection(false);
            activity6But.setSelection(false);
            activity7But.setSelection(false);
            activity8But.setSelection(false);
            activity9But.setSelection(false);

            if(!activity4But.getSelection())
            {
               redrawflag = 4;

            }
            else if(activity4But.getSelection())
            {
               redrawflag = 0;
            }

            flowgraphGroup.redraw();


            //Please Note:Checking for NOT condition because the
constBut button is RESET in one part of the code before we reach this point
            if(!activity4But.getSelection())
            {

   phaseAfterStart(myProp.getProperty("activity_4"),text,spinner);
            }
         }

      });


   //Refactoring action(button pressed) detected here
    activity5But.addMouseListener(new MouseAdapter() {
     public void mouseDown(MouseEvent e) {
            System.out.println("Refactoring Button Pressed");
            activity1But.setSelection(false);
            activity2But.setSelection(false);
            activity3But.setSelection(false);
            activity4But.setSelection(false);
            activity6But.setSelection(false);
            activity7But.setSelection(false);
            activity8But.setSelection(false);
            activity9But.setSelection(false);


            if(!activity5But.getSelection())
            {
               redrawflag = 5;

            }
            else if(activity5But.getSelection())
            {
               redrawflag = 0;
            }

            flowgraphGroup.redraw();

            //Please Note:Checking for NOT condition because the
refaBut button is RESET in one part of the code before we reach this point
            if(!activity5But.getSelection())
            {

   phaseAfterStart(myProp.getProperty("activity_5"),text,spinner);
            }

         }

      });


   //Review action(button pressed) detected here
    activity6But.addMouseListener(new MouseAdapter() {

     public void mouseDown(MouseEvent e) {
            System.out.println("Review Button Pressed");
            activity1But.setSelection(false);
            activity2But.setSelection(false);
            activity3But.setSelection(false);
            activity4But.setSelection(false);
            activity5But.setSelection(false);
            activity7But.setSelection(false);
            activity8But.setSelection(false);
            activity9But.setSelection(false);


            if(!activity6But.getSelection())
            {
               redrawflag = 6;
            }
            else if(activity6But.getSelection())
            {
               redrawflag = 0;
            }

            flowgraphGroup.redraw();


            //Please Note:Checking for NOT condition because the 
revBut button is RESET in one part of the code before we reach this point
            if(!activity6But.getSelection())
            {

   phaseAfterStart(myProp.getProperty("activity_6"),text,spinner);
            }
     }

      });


   //Integration Test action(button pressed) detected here
    activity7But.addMouseListener(new MouseAdapter() {

     public void mouseDown(MouseEvent e) {
            System.out.println("Integration Test Button 
Pressed");
            activity1But.setSelection(false);
            activity2But.setSelection(false);
            activity3But.setSelection(false);
            activity4But.setSelection(false);
            activity5But.setSelection(false);
            activity6But.setSelection(false);
            activity8But.setSelection(false);
            activity9But.setSelection(false);


            if(!activity7But.getSelection())
            {
               redrawflag = 7;

            }
            else if(activity7But.getSelection())
            {
               redrawflag = 0;
            }

            flowgraphGroup.redraw();


            //Please Note:Checking for NOT condition because the 
inttestBut button is RESET in one part of the code before we reach this point
            if(!activity7But.getSelection())
            {

   phaseAfterStart(myProp.getProperty("activity_7"),text,spinner);
            }
```

```
                  }

            });


      //Sandbox action(button pressed) detected here
        activity8But.addMouseListener(new MouseAdapter() {

          public void mouseDown(MouseEvent e) {
                        System.out.println("Sandbox Button Pressed");
                        activity1But.setSelection(false);
                        activity2But.setSelection(false);
                        activity3But.setSelection(false);
                        activity4But.setSelection(false);
                        activity5But.setSelection(false);
                        activity6But.setSelection(false);
                        activity7But.setSelection(false);
                        activity9But.setSelection(false);


                        if(!activity8But.getSelection())
                        {
                              redrawflag = 8;

                        }
                        else if(activity8But.getSelection())
                        {
                              redrawflag = 0;
                        }

                        flowgraphGroup.redraw();


                        //Please Note:Checking for NOT condition because the
sandBut button is RESET in one part of the code before we reach this point
                        if(!activity8But.getSelection())
                        {

      phaseAfterStart(myProp.getProperty("activity_8"),text,spinner);
                        }
                  }

            });


      //Postmortem action(button pressed) detected here
        activity9But.addMouseListener(new MouseAdapter() {

          public void mouseDown(MouseEvent e) {
                  System.out.println("Postmortem Button Pressed");

                        activity1But.setSelection(false);
                        activity2But.setSelection(false);
                        activity3But.setSelection(false);
                        activity4But.setSelection(false);
                        activity5But.setSelection(false);
                        activity6But.setSelection(false);
                        activity7But.setSelection(false);
                        activity8But.setSelection(false);


                        if(!activity9But.getSelection())
                        {
                              redrawflag = 9;

                        }
                        else if(activity9But.getSelection())
                        {
                              redrawflag = 0;
                        }

                        flowgraphGroup.redraw();


                        //Please Note:Checking for NOT condition because the
pmBut button is RESET in one part of the code before we reach this point
                        if(!activity9But.getSelection())
                        {

    phaseAfterStart(myProp.getProperty("activity_9"),text,spinner);
                        }
                  }
            });


          //Code Complete action(button pressed) detected here
          activity10But.addMouseListener(new MouseAdapter() {

           public void mouseDown(MouseEvent e) {
                        System.out.println("Code Complete Button Pressed");

           try {

      try{

            String command =
                  myProp.getProperty("process_runtime_command");
            Runtime rt = Runtime.getRuntime();
            Process pr = rt.exec(command);
            //Causing the current thread to sleep for
            5 sec, until the process pr completes
            Thread.sleep(5000);

            double ln_locbymethod = 0;
            double total_ln_locbymethod = 0;
            double Avg_ln_locbymethod = 0;
            double Std_ln_locbymethod = 0;
            int count = 0;
            double store_ln_locbymethod[] = new double[500];

                        //Reset piViewer(SizeMatrixView) index to 0;
                              SizeMatrixView.piviewerRowIndexCnt = 0;
      if(SizeMatrixView.piViewer.getElementAt(0) != null){
            SizeMatrixView.piViewer.getTable().clearAll();
      }

FileReader fr = new FileReader(
myProp.getProperty("sizeestimate_result_file") );
BufferedReader br = new BufferedReader(fr);
String s;
String szArray[] = null;


while((s = br.readLine()) != null)
{
      s = s.trim();
      szArray = s.split("  ");

      PartInformation piObj = new PartInformation();

      if(szArray.length >= 1)
      {
            piObj.setStrProjects(szArray[1]);
            piObj.setStrProxyName(szArray[2]);
            piObj.setStrLOC(szArray[3]);
            piObj.setStrNoOfMethods(szArray[4]);
            piObj.setStrRelSize(szArray[5]);
      }

      //Convert (LOC/METHODS) to LN(LOC/METHODS)
            ln_locbymethod = Math.log(Integer.parseInt(szArray[5]));

      //Storing LN(LOC/METHODS) in array
            store_ln_locbymethod[count] = ln_locbymethod;
            count = count + 1;

      //Sum LN(LOC/METHODS)
            total_ln_locbymethod = total_ln_locbymethod + ln_locbymethod;

      setSelection(new StructuredSelection(piObj));
      }

      //Reset smViewer(SizeMatrixView) index to 0;
      SizeMatrixView.smviewerRowIndexCnt = 0;
if(SizeMatrixView.smViewer.getElementAt(0) != null)
      {

            SizeMatrixView.smViewer.getTable().clearAll();
      }

      //Calculate AVERAGE(LN(LOC/METHODS))
      Avg_ln_locbymethod = (total_ln_locbymethod / count);
      System.out.println("AVERAGE LN(LOC/METHODS) is: "+Avg_ln_locbymethod);

      //Calculate STDEV(LN(LOC/METHODS))
      // calculating the sum of squares
            double sum = 0;
            for ( int i=0; i<count; i++ )
            {
```

```
    final double v = store_ln_locbymethod[i] - Avg_ln_locbymethod;
    sum += v * v;
 }

 Std_ln_locbymethod = Math.sqrt(sum/count);
 System.out.println("STDEV LN(LOC/METHODS) is:
       "+Std_ln_locbymethod);

 //Calculate VS Low(1) Mid(=CEILING(EXP(Average-2*StdDev),1))
 High(=CEILING(EXP(Average-1.5*StdDev),1))
 //VS Low(1)
 //VS Mid
 double VSMid = Math.ceil(Math.exp(Avg_ln_locbymethod-
                         (2*Std_ln_locbymethod)));
 //VS High
 double VSHigh = Math.ceil(Math.exp(Avg_ln_locbymethod-
                         (1.5*Std_ln_locbymethod)));

 SizeMatrix smObj = new SizeMatrix();
       smObj.setStrBucket("  VS  ");
       smObj.setStrLow("1");
       smObj.setStrMid(Double.toString(VSMid));

       smObj.setStrHigh(Double.toString(VSHigh));
                   setSelection(new StructuredSelection(smObj));


 //Calculate S Low(=CEILING(EXP(Average-1.5*StdDev),1))
 Mid(=CEILING(EXP(Average-StdDev),1))
 High(=CEILING(EXP(Average-0.5*StdDev),1))
 //S Low
 double SLow = Math.ceil(Math.exp(Avg_ln_locbymethod-
                                     (1.5*Std_ln_locbymethod)));
 //S Mid
 double SMid = Math.ceil(Math.exp(Avg_ln_locbymethod-
                                           Std_ln_locbymethod));
 //S High
 double SHigh = Math.ceil(Math.exp(Avg_ln_locbymethod-
                                     (0.5*Std_ln_locbymethod)));

 smObj.setStrBucket("  S  ");
 smObj.setStrLow(Double.toString(SLow));
 smObj.setStrMid(Double.toString(SMid));
 smObj.setStrHigh(Double.toString(SHigh));
 setSelection(new StructuredSelection(smObj));


//Calculate M Low(=CEILING(EXP(Average-0.5*StdDev),1))
 Mid(=CEILING(EXP(Average),1))
 High(=CEILING(EXP(Average+0.5*StdDev),1))
 //M Low
 double MLow = Math.ceil(Math.exp(Avg_ln_locbymethod-
                               (0.5*Std_ln_locbymethod)));
 //M Mid
 double MMid = Math.ceil(Math.exp(Avg_ln_locbymethod));
 //M High
 double MHigh =
 Math.ceil(Math.exp(Avg_ln_locbymethod+(0.5*Std_ln_locbymethod)));

 smObj.setStrBucket("  M  ");
 smObj.setStrLow(Double.toString(MLow));
 smObj.setStrMid(Double.toString(MMid));
 smObj.setStrHigh(Double.toString(MHigh));
 setSelection(new StructuredSelection(smObj));


 //Calculate L Low(=CEILING(EXP(Average+0.5*StdDev),1))
 Mid(=CEILING(EXP(Average+StdDev),1))
 High(=CEILING(EXP(Average+1.5*StdDev),1))
 //L Low
 double LLow =
 Math.ceil(Math.exp(Avg_ln_locbymethod+(0.5*Std_ln_locbymethod)));
 //L Mid
 double LMid =
 Math.ceil(Math.exp(Avg_ln_locbymethod+Std_ln_locbymethod));
 //L High
 double LHigh =
 Math.ceil(Math.exp(Avg_ln_locbymethod+(1.5*Std_ln_locbymethod)));

 smObj.setStrBucket("  L  ");
 smObj.setStrLow(Double.toString(LLow));
 smObj.setStrMid(Double.toString(LMid));
 smObj.setStrHigh(Double.toString(LHigh));
 setSelection(new StructuredSelection(smObj));


 //Calculate VL Low(=CEILING(EXP(Average+1.5*StdDev),1))
 Mid(=CEILING(EXP(Average+2*StdDev),1)) High(big))
 //VL Low
 double VLLow =
 Math.ceil(Math.exp(Avg_ln_locbymethod+(1.5*Std_ln_locbymethod)))
 //VL Mid
 double VLMid =
 Math.ceil(Math.exp(Avg_ln_locbymethod+(2*Std_ln_locbymethod)));
 //VL High (big)


 smObj.setStrBucket("  VL  ");
 smObj.setStrLow(Double.toString(VLLow));
 smObj.setStrMid(Double.toString(VLMid));
 smObj.setStrHigh("Big");
 setSelection(new StructuredSelection(smObj));


 }catch (IOException ie) {
       ie.printStackTrace();
 }

 } catch (Exception e1) {
       e1.printStackTrace();
 }


 if(!activity10But.getSelection())
```

```
            {
                  redrawflag = 10;

            }
            else if(activity10But.getSelection())
            {
                  redrawflag = 0;
            }

            flowgraphGroup.redraw();

          }
            });


        startstopBut.addMouseListener(new MouseAdapter() {
                @Override
                public void mouseDown(MouseEvent e) {



                  //Validations
      if(!activity1But.getSelection() && !activity2But.getSelection() &&
!activity3But.getSelection() && !activity4But.getSelection() &&
!activity5But.getSelection() && !activity6But.getSelection() &&
!activity7But.getSelection() && !activity8But.getSelection() &&
!activity9But.getSelection() )

        {
            //Java Swing dialog box
            JOptionPane.showMessageDialog(null,
            "1.) Please select a Phase.","Project Dashboard Warning",
                      JOptionPane.WARNING_MESSAGE);

                  startstopBut.setSelection(true);
        }
        else if(activity1But.getSelection() || activity2But.getSelection() ||
activity3But.getSelection() || activity4But.getSelection() ||
activity5But.getSelection() || activity6But.getSelection() ||
activity7But.getSelection() || activity8But.getSelection() ||
activity9But.getSelection() && !text.getText().trim().equals(""))
        {
            //STOP Logic goes here
            if(intStartButPressFlag == 1)
            {
                  intStartButPressFlag = 0;
                  intPhaseAfterStartFlag = 0;

      globalTlg.setIntStartStopButPressFlag(intStartButPressFlag);

      globalTlg.setIntPhaseAfterStartButPressFlag(intPhaseAfterStartFlag);
                  startstopBut.setText("       START        ");

                  //Getting & Setting the Stop time
                  Calendar cal = Calendar.getInstance();
                  SimpleDateFormat sdfTime = new SimpleDateFormat("h:mm a");
                  sdfTime.format(cal.getTime());
            globalTlg.setStoptime(sdfTime.format(cal.getTime()));


            long lgstopTime = System.currentTimeMillis();
            globalTlg.setCalcStoptime(lgstopTime);


            long lgstartTime = globalTlg.getCalcStarttime();


            long diff = lgstopTime - lgstartTime;

            long diffMinutes = diff / (60 * 1000);
            long diffHours = diff / (60 * 60 * 1000);

            String strDelta = new String(diffHours+" hr "+diffMinutes+"
            min ");
            globalTlg.setDelta(strDelta);


            setSelection(new StructuredSelection(globalTlg));

            }
            //START Logic goes here
            else if(intStartButPressFlag == 0)
            {
                  Timelog tlg = new Timelog();

                  //Setting the Phase
                  if(activity1But.getSelection())
                  tlg.setPhase( myProp.getProperty("activity_1") );
                  else if(activity2But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_2") );
            else if(activity3But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_3") );
            else if(activity4But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_4") );
            else if(activity5But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_5") );
            else if(activity6But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_6") );
            else if(activity7But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_7") );
            else if(activity8But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_8") );
            else if(activity9But.getSelection())
            tlg.setPhase( myProp.getProperty("activity_9") );

            //Setting the Comments


tlg.setComments(text.getText().trim());

//Setting the Feature Set
tlg.setFeatureSet(Integer.toString(spinner.getSelection()));
//Getting & Setting the Date and Time
```

```
      Calendar cal = Calendar.getInstance();

      SimpleDateFormat sdfDate = new SimpleDateFormat("MM/dd/yy");
       sdfDate.format(cal.getTime());
      tlg.setDate(sdfDate.format(cal.getTime()));

      SimpleDateFormat sdfTime = new SimpleDateFormat("h:mm a");
                              sdfTime.format(cal.getTime());

      tlg.setStarttime(sdfTime.format(cal.getTime()));

      long lgTime = System.currentTimeMillis();
      tlg.setCalcStarttime(lgTime);

      intStartButPressFlag = 1;

      tlg.setIntStartStopButPressFlag(intStartButPressFlag);
                              startstopBut.setText("        STOP         ");

      //Assigning the local tlg object to global timelog object, inorder to
      //refer to the same entry and calculate stop time, delta when STOP
      TIMER is pressed
       globalTlg = tlg;


      setSelection(new StructuredSelection(globalTlg));

      //Enabling the flag for PHASE AFTER START
      intPhaseAfterStartFlag = 1;

      tlg.setIntPhaseAfterStartButPressFlag(intPhaseAfterStartFlag);

      globalTlg.setIntStartStopButPressFlag(intPhaseAfterStartFlag);
                              }
                        }
                ]
               });
      }

      private class FlowGraphPaintListener implements PaintListener {
        public void paintControl(PaintEvent e) {

          //Load the property file before use
          myProp = loadProperties(PROPFILE);

          e.gc.setForeground(e.display.getSystemColor(SWT.COLOR_BLACK));
          // Set the width of the lines to draw
              e.gc.setLineWidth(2);

              if(redrawflag == 0)
              {
                    //set background color

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

      //Draw Rectangle
      e.gc.drawRectangle(130,12,20,20);
```

```
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);


 //Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);

//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);

//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);

//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);
```

```
//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
            }
else if (redrawflag == 1)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);

//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);
```

```
//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);

//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);

//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

//Draw Line
e.gc.drawLine(140, 217, 140, 227);
//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);
```

```
//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);

}
else if (redrawflag == 2)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));
//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);

//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);

//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);
```

```
//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
else if (redrawflag == 3)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);
```

```
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);

//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);
//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);
```

```
//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
else if (redrawflag == 4)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));
```

```
//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
            e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);

//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);

//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);
//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
else if (redrawflag == 5)
```

```
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);


//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);

//Draw Line
e.gc.drawLine(140, 167, 140, 172);
```

```
//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
else if (redrawflag == 6)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
```

```
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);


//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);


//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));
//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);
```

```
//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
else if (redrawflag == 7)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);


//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);
```

```
//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);


//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
```

```
else if (redrawflag == 8)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);


//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);

//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);


//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);


//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
```

```
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
else if (redrawflag == 9)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);

//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
```

```
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);


//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);


//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);


//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);


//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);


//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 217, 140, 227);
```

```
//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);

//Draw Vz connection line
e.gc.drawLine(90, 87, 90, 252);

//Draw Rectangle
e.gc.drawRectangle(130,257,20,20);
e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
}
else if (redrawflag == 10)
{
//set background color
e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_WHITE));

//Draw Rectangle
e.gc.drawRectangle(130,12,20,20);
e.gc.drawText(myProp.getProperty("activity_1_symbol"), 132, 14);
//Draw Line
e.gc.drawLine(140, 32, 140, 37);

//Draw Rectangle
e.gc.drawRectangle(130,37,20,20);
e.gc.drawText(myProp.getProperty("activity_2_symbol"), 132, 39);

//Draw Line
e.gc.drawLine(140, 57, 140, 62);

//Draw Rectangle
e.gc.drawRectangle(130,62,20,20);
e.gc.drawText(myProp.getProperty("activity_3_symbol"), 132, 64);


//Draw Line
e.gc.drawLine(140, 82, 140, 92);

//Draw Hz connection line
e.gc.drawLine(90, 87, 140, 87);

//Draw Rectangle
e.gc.drawRectangle(130,92,20,20);
e.gc.drawText(myProp.getProperty("activity_4_symbol"), 132, 94);
```

```
//Draw Line
e.gc.drawLine(140, 112, 140, 122);

//Draw Hz connection line
e.gc.drawLine(110, 117, 140, 117);

//Draw Rectangle
e.gc.drawRectangle(130,122,20,20);
e.gc.drawText(myProp.getProperty("activity_5_symbol"), 132, 124);


//Draw Line
e.gc.drawLine(140, 142, 140, 147);

//Draw Rectangle
e.gc.drawRectangle(130,147,20,20);
e.gc.drawText(myProp.getProperty("activity_6_symbol"), 132, 149);


//Draw Line
e.gc.drawLine(140, 167, 140, 172);

//Draw Rectangle
e.gc.drawRectangle(130,172,20,20);
e.gc.drawText(myProp.getProperty("activity_7_symbol"), 134, 174);


//Draw Line
e.gc.drawLine(140, 192, 140, 197);

//Draw Rectangle
e.gc.drawRectangle(130,197,20,20);
e.gc.drawText(myProp.getProperty("activity_8_symbol"), 132, 199);


//Draw Line
e.gc.drawLine(140, 217, 140, 227);

//Draw Hz connection line
e.gc.drawLine(110, 222, 140, 222);

//Draw Vz connection line
e.gc.drawLine(110, 117, 110, 222);

//Draw Rectangle
e.gc.drawRectangle(130,227,20,20);
e.gc.drawText(myProp.getProperty("activity_9_symbol"), 132, 229);

e.gc.setBackground(e.display.getSystemColor(SWT.COLOR_GREEN));

//Draw Line
e.gc.drawLine(140, 247, 140, 257);

//Draw Hz connection line
e.gc.drawLine(90, 252, 140, 252);
```

```
      //Draw Vz connection line
      e.gc.drawLine(90, 87, 90, 252);

      //Draw Rectangle
      e.gc.drawRectangle(130,257,20,20);
      e.gc.drawText(myProp.getProperty("activity_10_symbol"), 132, 259);
      }

      }
      }

      public void phaseAfterStart(String phase,Text text, Spinner spinner)
      {
            if( (intStartButPressFlag == 1) && (intPhaseAfterStartFlag == 1)
)
            {
                  //LOGIC TO COMPLETE THE EXISTING RECORD
                  //Getting & Setting the Stop time
                  Calendar cal = Calendar.getInstance();
                  SimpleDateFormat sdfTime = new SimpleDateFormat("h:mm a");
            sdfTime.format(cal.getTime());
          System.out.println("The Stop Time is:
"+sdfTime.format(cal.getTime()));
            globalTlg.setStoptime(sdfTime.format(cal.getTime()));


                  long lgstopTime = System.currentTimeMillis();
            globalTlg.setCalcStoptime(lgstopTime);


            long lgstartTime = globalTlg.getCalcStarttime();
                  long diff = lgstopTime - lgstartTime;

                  long diffMinutes = diff / (60 * 1000);
                  long diffHours = diff / (60 * 60 * 1000);

                  String strDelta = new String(diffHours+" hr "+diffMinutes+"
min ");
                  globalTlg.setDelta(strDelta);

                  setSelection(new StructuredSelection(globalTlg));


                  //LOGIC TO INSERT NEW RECORD
                  Timelog newtlg = new Timelog();

                  //Setting the FLAG by to retain the LAST STATE of the
TableViewer

      newtlg.setIntStartStopButPressFlag(globalTlg.getIntStartStopButPressFla
g());

      newtlg.setIntPhaseAfterStartButPressFlag(globalTlg.getIntPhaseAfterStar
tButPressFlag());
```

```
                  //Setting the FLAG to 1 for inserting the row correctly
during the PHASE AFTER START
                  //newtlg.intAddlnPhaseRowsFlag = 1;

      //newtlg.setIntAddlnPhaseRowsFlag(newtlg.intAddlnPhaseRowsFlag);
                  newtlg.setIntAddlnPhaseRowsFlag(1);

                  //Setting the Phase
                  if(phase.equals( myProp.getProperty("activity_1") ))
                        newtlg.setPhase( myProp.getProperty("activity_1") );
                  else if(phase.equals( myProp.getProperty("activity_2") ))
                        newtlg.setPhase( myProp.getProperty("activity_2") );
                  else if(phase.equals( myProp.getProperty("activity_3") ))
                        newtlg.setPhase( myProp.getProperty("activity_3") );
                  else if(phase.equals( myProp.getProperty("activity_4") ))
                        newtlg.setPhase( myProp.getProperty("activity_4") );
                  else if(phase.equals( myProp.getProperty("activity_5") ))
                        newtlg.setPhase( myProp.getProperty("activity_5") );
                  else if(phase.equals( myProp.getProperty("activity_6") ))
                        newtlg.setPhase( myProp.getProperty("activity_6") );
                  else if(phase.equals( myProp.getProperty("activity_7") ))
                        newtlg.setPhase( myProp.getProperty("activity_7") );
                  else if(phase.equals( myProp.getProperty("activity_8") ))
                        newtlg.setPhase( myProp.getProperty("activity_8") );
                  else if(phase.equals( myProp.getProperty("activity_9") ))
                        newtlg.setPhase( myProp.getProperty("activity_9") );


                  //Setting the Comments
                  newtlg.setComments(text.getText().trim());

                  //Setting the Feature Set
      newtlg.setFeatureSet(Integer.toString(spinner.getSelection()));


                  //Getting & Setting the Date and Time
                  Calendar newcal = Calendar.getInstance();

                  SimpleDateFormat newsdfDate = new
SimpleDateFormat("MM/dd/yy");
                  newsdfDate.format(newcal.getTime());
                  newtlg.setDate(newsdfDate.format(cal.getTime()));

                  //Setting the START TIME as the STOP TIME of the previous
                  row/record.
            newtlg.setStarttime(globalTlg.getStoptime());
            newtlg.setCalcStarttime(globalTlg.getCalcStoptime());

            //Assigning the local tlg object to global timelog object,
            inorder to
            //refer to the same entry and calculate stop time, delta when
            STOP TIMER is pressed
            globalTlg = newtlg;

            setSelection(new StructuredSelection(globalTlg));

            //After inserting setting back the FLAG to 0 for STOP logic
            during the PHASE AFTER START
            newtlg.setIntAddlnPhaseRowsFlag(0);
                  globalTlg.setIntAddlnPhaseRowsFlag(0);

                  return;
            }
      }

      private void fillLocalPullDown(IMenuManager manager) {
            manager.add(action1);
            manager.add(new Separator());
            manager.add(action2);
      }

      /**
       * Passing the focus request to the form.
       */
      public void setFocus() {
            form.setFocus();
      }

      /**
       * Disposes the toolkit
       */
      public void dispose() {
            toolkit.dispose();
            super.dispose();
      }

      //ISelectionProvider abstract methods
      @Override
      public void addSelectionChangedListener(ISelectionChangedListener
listener) {
            listeners.add(listener);
      }

      @Override
      public ISelection getSelection() {
             return null;
             //return new StructuredSelection(selection);
      }

      @Override
      public void removeSelectionChangedListener(
                  ISelectionChangedListener listener) {
              listeners.remove(listener);
      }

      @Override
      public void setSelection(ISelection selection) {
             Object[] list = listeners.getListeners();
               for (int i = 0; i < list.length; i++) {
                ((ISelectionChangedListener) list[i])
                  .selectionChanged(new SelectionChangedEvent(this,
selection));
               }
      }


      //Function to load the properties file
      private static Properties loadProperties(String fileName)
      {
            InputStream propsFile;
        Properties tempProp = new Properties();

        try
        {
            propsFile = new FileInputStream(fileName);
            tempProp.load(propsFile);
            propsFile.close();
        }catch (IOException ioe) {
            System.out.println("I/O Exception.");
            ioe.printStackTrace();
            System.exit(0);
        }

        return tempProp;

    }

}
```

FileBrowse.java

```
/*
 * FileBrowse class is used to open the file( i.e. File Open)
 */
package org.example.tableviewer.views;
import java.awt.Container;
import java.awt.FileDialog;
import java.awt.GridLayout;
import java.io.File;
import javax.swing.JFrame;


public class FileBrowse extends JFrame {

      private static final long serialVersionUID = 1L;
      private String directoryFile = "";

    public FileBrowse() {
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setResizable(false);

                onBrowseNative();
        Container container = getContentPane();
        container.setLayout(new GridLayout(0, 1));

    }


    @SuppressWarnings("deprecation")
      protected void onBrowseNative() {
        FileDialog fileDialog = new FileDialog(this, "Open/Save");
        fileDialog.~~show~~();
        if (fileDialog.getFile() != null) {
            String directory = fileDialog.getDirectory();
            if (!directory.endsWith(File.separator)) {
                directory += File.separator;
            }
            String file = fileDialog.getFile();
            directoryFile = directory + file;
        }
    }

    public String getDirectoryFile() {
            return directoryFile;
      }

      public void setDirectoryFile(String directoryFile) {
            this.directoryFile = directoryFile;
      }

}
```

DisplayRepositoryTree.java

```
/*
 * DisplayRepositoryTree will connect to Apache Subversion, read
 * .java files and retrieves parts information (LOC, No. of Methods,
 * Inner/Outer class, Relative size) and write it to a text file
 */

package org.example.tableviewer.views;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.Authenticator;
import java.net.PasswordAuthentication;
import java.net.URL;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.Properties;
import org.tmatesoft.svn.core.*;
import org.tmatesoft.svn.core.auth.ISVNAuthenticationManager;
import org.tmatesoft.svn.core.internal.io.dav.DAVRepositoryFactory;
import org.tmatesoft.svn.core.internal.io.fs.FSRepositoryFactory;
import org.tmatesoft.svn.core.internal.io.svn.SVNRepositoryFactoryImpl;
import org.tmatesoft.svn.core.io.SVNRepository;
import org.tmatesoft.svn.core.io.SVNRepositoryFactory;
import org.tmatesoft.svn.core.wc.SVNWCUtil;
public class DisplayRepositoryTree
{
      static BufferedWriter out = null;

      static String PROPFILE=
"C:/Users/pzs0011/workspace/org.example.TableViewer/attributes.properties";
      static Properties myProp;

    public static void main(String[] args)
    {

      //Load the property file
      myProp = loadProperties(PROPFILE);

      String url = myProp.getProperty("svn_url");
      String name = myProp.getProperty("svn_username");
      String password = myProp.getProperty("svn_password");

        setupLibrary();

        SVNRepository repository = null;
        try
        {
/*
* Creates an instance of SVNRepository to work with the repository.
* All user's requests to the repository are relative to the
* repository location used to create this SVNRepository.
* SVNURL is a wrapper for URL strings that refer to repository locations.
*/
           repository =
                  SVNRepositoryFactory.create(SVNURL.parseURIEncoded(url));
        }
        catch (SVNException svne)
        {
            /*
             * Perhaps a malformed URL is the cause of this exception
             */
            System.err.println("error while creating an SVNRepository for
                        location '"+ url + "': " + svne.getMessage());
            System.exit(1);
        }

/*
* User's authentication information (name/password) is provided via  an
* ISVNAuthenticationManager  instance.  SVNWCUtil  creates  a   default
* authentication manager given user's name and password.
*
* Default authentication manager first attempts to use provided user name
* and password and then falls back to the credentials stored in the
* default Subversion credentials storage that is located in Subversion
* configuration area. If you'd like to use provided user name and password
* only you may use BasicAuthenticationManager class instead of default
* authentication manager:
*
*  authManager = new BasicAuthenticationsManager(userName, userPassword);
*
* You may also skip this point - anonymous access will be used.
*/
        ISVNAuthenticationManager authManager =
SVNWCUtil.createDefaultAuthenticationManager(name, password);
        repository.setAuthenticationManager(authManager);

        try
        {
/*
* Checks up if the specified path/to/repository part of the URL
* really corresponds to a directory. If doesn't the program exits.
* SVNNodeKind is that one who says what is located at a path in a
* revision. -1 means the latest revision.
*/
 SVNNodeKind nodeKind = repository.checkPath("", -1);
 if (nodeKind == SVNNodeKind.NONE)
 {
      System.err.println("There is no entry at '" + url + "'.");
      System.exit(1);
 }
else if (nodeKind == SVNNodeKind.FILE)
{
     System.err.println("The entry at '" + url + "' is a file while a
directory was expected.");
     System.exit(1);
}

        System.out.println("The Results are :(Format ::
OUTER_CLASS/INNER_CLASS  PROJECT  PROXY_NAME  NO_OF_LINES  NO_OF_METHODS
LOC/METHOD)");

 //Writing to File.
 FileWriter fstream;
 try
 {
      fstream = new FileWriter(
                        myProp.getProperty("sizeestimate_result_file") );
                        out = new BufferedWriter(fstream);
 }
 catch (IOException e)
 {
      e.printStackTrace();
 }

/*
 * Displays the repository tree at the current path - "" (what means
 * the path/to/repository directory)
 */
 listEntries(repository, "");
```

```
      try
      {
            out.close();
      } catch (IOException e)
      {
            e.printStackTrace();
      }

 }
  catch (SVNException svne)
  {
       System.err.println("error while listing entries: "
                 + svne.getMessage());
       System.exit(1);
  }
}//End of Main



    /*
     * Initializes the library to work with a repository via
     * different protocols.
     */
    private static void setupLibrary()
    {
        /*
         * For using over http:// and https://
         */
        DAVRepositoryFactory.setup();
        /*
         * For using over svn:// and svn+xxx://
         */
        SVNRepositoryFactoryImpl.setup();

        /*
         * For using over file:///
         */
        FSRepositoryFactory.setup();
    }



    /*
* Called recursively to obtain all entries that make up the repository tree
* repository - an SVNRepository which interface is used to carry out the
* request, in this case it's a request to get all entries in the directory
* located at the path parameter;
*
* path is a directory path relative to the repository location path (that
* is a part of the URL used to create an SVNRepository instance);
*
*/
    public static void listEntries(SVNRepository repository, String path)
            throws SVNException
    {
/*
```

```
* Gets the contents of the directory specified by path at the latest
* revision (for this purpose -1 is used here as the revision number to
* mean HEAD-revision) getDir returns a Collection of SVNDirEntry
* elements. SVNDirEntry represents information about the directory
* entry. Here this information is used to get the entry name, the name
* of the person who last changed this entry, the number of the revision
* when it was last changed and the entry type to determine whether it's
* a directory or a file. If it's a directory listEntries steps into a
* next recursion to display the contents of this directory. The third
* parameter of getDir is null and means that a user is not interested
* in directory properties. The fourth one is null, too - the user
* doesn't provide its own Collection instance and uses the one returned
* by getDir.
*/


       Collection entries = repository.getDir(path, -1, null,
               (Collection) null);
       Iterator iterator = entries.iterator();
       while (iterator.hasNext())
       {
           SVNDirEntry entry = (SVNDirEntry) iterator.next();

           //Code to Filter Directory
           if(!(entry.getKind() == SVNNodeKind.DIR))
              {
                      //strPath stores the full directory path (for e.g.
                       Project3/DisplayRepositoryTree.java)
           String strPath = (path.equals("")) ? entry.getName()
                               : path + "/" + entry.getName();


     try
     {
           final String login = myProp.getProperty("svn_username");
           final String password = myProp.getProperty("svn_password");

           Authenticator.setDefault(new Authenticator()
           {
                 protected PasswordAuthentication
                                   getPasswordAuthentication()
                 {
                       return new PasswordAuthentication (login,
                                         password.toCharArray());
           }
           });

           URL svn = new URL(myProp.getProperty("svn_url")+strPath);


           BufferedReader in = new BufferedReader(
                     new InputStreamReader(svn.openStream()));

                 String inputLine = "";

                 String classArray[] = null;
                 String defclassArray[] = null;
```

```
                  String methodArray[] = null;
                  String commentArray[] = null;

                  ArrayList<String> finalClassArr = new ArrayList<String>();
                  ArrayList<String> finalDefClassArr = new
                                                    ArrayList<String>();

                  boolean commentflag = false;

                  int oCnt = 0; //Outer Class Index for finalClassArr
                  int iCnt = 0; //Inner Class Index for finalDefClassArr

                  int omCnt = 0; //Outer Method Counter
                  int imCnt = 0; //Inner Method Counter

                  int lineCounter = 0;
                  int icLineCounter = 0;
                  int openBrCounter = 0;
                  int closeBrCounter = 0;
                  int icOpenBrCounter = 0;
                  int icCloseBrCounter = 0;
                  boolean lineCounterFlag = false;
                  boolean icFlag = false;

                  String projNameArray[] = null;
                  projNameArray = strPath.split("/");

                  String fileName = projNameArray[projNameArray.length - 1];

                  //Allowing only java files
                  if(fileName.contains(".java"))
                  {
                        while ((inputLine = in.readLine()) != null)
                        {
                            inputLine = inputLine.trim();
                            commentArray = inputLine.split("");
                              if( (commentArray.length >= 2) &&
(commentArray[commentArray.length - 1].contentEquals("/")) &&
(commentArray[commentArray.length - 2].contentEquals("*")) )
            {
                        commentflag = false;
            }

            if( (inputLine.trim().length() != 0) && (commentflag == false) )
            {

            if( (commentArray.length >= 2) &&
(commentArray[1].contentEquals("/")) && (commentArray[2].contentEquals("*"))
)
            {
                                                        commentflag = true;

            }
            else if( (commentArray.length >= 2) &&
commentArray[1].contentEquals("/") && commentArray[2].contentEquals("/") )
            {
                  //"//" comment triggered
                  //Skip this line
            }
            else if( (commentArray.length >= 2) &&
                        commentArray[1].contentEquals("@") )
            {
                  //annotation @ triggered
                  //skip this line
            }
            else if( (commentArray.length >= 2) &&
(commentArray[commentArray.length - 1].contentEquals("/")) &&
(commentArray[commentArray.length - 2].contentEquals("*")) )
            {
                  commentflag = false;
            }
            else
            {

                  if( (inputLine.contains("private class ")) ||
(inputLine.contains("public class ")) || (inputLine.contains("protected class
")) || (inputLine.contains("default class ")) || (inputLine.contains("class
")) )
                  {
                        if( !(inputLine.contains("\"")) &&
!(inputLine.contains("/*")) && !(inputLine.contains("*/")) &&
!(inputLine.contains("*")) && !(inputLine.contains("//")) )
                        {
                              //Parsing logic to get the outer classes

if(openBrCounter == closeBrCounter)
{
            lineCounterFlag = true;
                  classArray = inputLine.split(" ");
            int index = 0;

      if( (classArray.length >= 3) &&
                                          (classArray[2].contains("{")) )
      {

            index = classArray[2].indexOf("{");
            classArray[2] = classArray[2].substring(0, index);
      }
      else if( (classArray.length >= 2) && (classArray[1].contains("{")) )
      {

              index = classArray[1].indexOf("{");

              classArray[1] = classArray[1].substring(0, index);
      }

      if( (inputLine.contains("private class ")) ||
(inputLine.contains("public class ")) || (inputLine.contains("protected class
")) || (inputLine.contains("default class ")) )
      {

              finalClassArr.add(classArray[2]);
      }
      else if(inputLine.contains("class "))
      {
            finalClassArr.add(classArray[1]);
      }
}
//Parsing logic to handle default/inner class including inner classes with
public, protected and private access modifiers
else if(openBrCounter != closeBrCounter)
{

icFlag = true;
defclassArray = inputLine.split(" ");

int index = 0;
if( (defclassArray.length >= 3) && (defclassArray[2].contains("{")) )
{
      index = defclassArray[2].indexOf("{");
      defclassArray[2] = defclassArray[2].substring(0, index);
}
else if( (defclassArray.length >= 2) && (defclassArray[1].contains("{")) )
{
      index = defclassArray[1].indexOf("{");
      defclassArray[1] = defclassArray[1].substring(0, index);
}

if( (inputLine.contains("private class ")) || (inputLine.contains("public
class ")) || (inputLine.contains("protected class ")) ||
(inputLine.contains("default class ")) )
{

            finalDefClassArr.add(defclassArray[2]);
}
else if(inputLine.contains("class "))
{
      finalDefClassArr.add(defclassArray[1]);
}


}

}

}




if(lineCounterFlag)
{
      lineCounter = lineCounter + 1;

      if( (inputLine.charAt(inputLine.length()-1) == '{') )
      {
              openBrCounter = openBrCounter + 1;
      }

      if( (inputLine.length() >= 2) && !(inputLine.contains("{")) &&
            (inputLine.charAt(inputLine.length()-2) == '}') &&
            (inputLine.charAt(inputLine.length()-1) == ';') )
      {

            closeBrCounter = closeBrCounter + 1;
      }
      else if( (inputLine.length() >= 1) &&
                        (inputLine.charAt(inputLine.length()-1) == '}') )
      {
            closeBrCounter = closeBrCounter + 1;
      }
      else if(inputLine.charAt(0) == '}')
      {
            closeBrCounter = closeBrCounter + 1;
      }


  //Parsing logic to get the methods from outer class
      methodArray = inputLine.split(" ");

      if( (inputLine.contains("private ")) || (inputLine.contains("public "))
|| (inputLine.contains("protected ")) || (inputLine.contains("default ")) &&
!(inputLine.contains("class ")))
      {

      if(!(inputLine.contains("\"")) && (inputLine.contains("(")) &&
      (inputLine.contains(")")) )
      {

      //Outer Methods
      if( (methodArray.length >= 3) && !(methodArray[1].contains("(")))
      {
              omCnt = omCnt + 1;
      }
      //To consider methods with no access modifiers
      else if( (methodArray.length >= 2) && (methodArray[1].contains("(")))
      {
            omCnt = omCnt + 1;
      }
    }
  }

}




//Print outer class count
if( (openBrCounter > 0) && (closeBrCounter > 0) && (openBrCounter ==
closeBrCounter) )
{
      //Calculate No. of lines/Methods
      int lnbymethod = 0;

      if(omCnt > 0)
      {
            lnbymethod = lineCounter/omCnt;
      }
      else
      {
          lnbymethod = lineCounter;
      }

      System.out.println("O  "+projNameArray[0]+"
"+finalClassArr.get(oCnt)+"  "+lineCounter+"  "+omCnt+"  "+lnbymethod);
out.write("O  "+projNameArray[0]+"  "+finalClassArr.get(oCnt)+"
"+lineCounter+"  "+omCnt+"  "+lnbymethod);
out.newLine();
      oCnt++;

      omCnt = 0;
      lineCounter = 0;
      openBrCounter = 0;
      closeBrCounter = 0;
      lineCounterFlag = false;


}


//Start inner class counter
if(icFlag == true)
{
      icLineCounter = icLineCounter + 1;

      if( (inputLine.charAt(inputLine.length()-1) == '{') )
      {
            icOpenBrCounter = icOpenBrCounter + 1;
      }

      if( (inputLine.length() >= 2) && !(inputLine.contains("{")) &&
                  (inputLine.charAt(inputLine.length()-2) == '}') &&
            (inputLine.charAt(inputLine.length()-1) == ';') )
      {
            closeBrCounter = closeBrCounter + 1;
      }
      else if( (inputLine.length() >= 1) &&
                  (inputLine.charAt(inputLine.length()-1) == '}') )
      {
            icCloseBrCounter = icCloseBrCounter + 1;
      }
      else if(inputLine.charAt(0) == '}')
      {
            icCloseBrCounter = icCloseBrCounter + 1;
      }



      //Parsing logic to get the methods from default/inner class
      methodArray = inputLine.split(" ");

      if( (inputLine.contains("private ")) || (inputLine.contains("public "))
|| (inputLine.contains("protected ")) || (inputLine.contains("default ")) &&
!(inputLine.contains("class ")) )
      {
```

```
if(!(inputLine.contains("\"")) && (inputLine.contains("(")) &&
(inputLine.contains(")")) )
      {

//Inner Methods
if( (methodArray.length >= 3) && !(methodArray[1].contains("(")))
{

              imCnt = imCnt + 1;


}

//To consider methods with no access modifiers

else if( (methodArray.length >= 2) && (methodArray[1].contains("(")))
{

              imCnt = imCnt + 1;
}
}
}
}


//Print inner class count
if((icOpenBrCounter > 0) && (icCloseBrCounter > 0) && (icOpenBrCounter ==
icCloseBrCounter))
      {
      //Calculate No. of lines/Methods
      int lnbymethod = 0;

      if(imCnt > 0)
      {
            lnbymethod = icLineCounter/imCnt;
      }
      else
      {
            lnbymethod = icLineCounter;
      }


System.out.println("I  "+projNameArray[0]+"  "+finalDefClassArr.get(iCnt)+"
"+icLineCounter+"  "+imCnt+"  "+lnbymethod);
out.write("I  "+projNameArray[0]+"  "+finalDefClassArr.get(iCnt)+"
"+icLineCounter+"  "+imCnt+"  "+lnbymethod);
out.newLine();
iCnt++;

imCnt = 0;
icLineCounter = 0;
icOpenBrCounter = 0;

icCloseBrCounter = 0;
icFlag = false;
```

```
      }
   }
}

}//end while
in.close();
}

}
catch(Exception e)
{
      e.printStackTrace();
}

}


/*
* Checking up if the entry is a directory.
* Recursive call to function listEntries
*/
      if (entry.getKind() == SVNNodeKind.DIR)
       {
             listEntries(repository, (path.equals("")) ? entry.getName()
                        : path + "/" + entry.getName());
                  }
        }//End while
    }//End listEntries
    /** Get the current line number.
     * @return int - Current line number.
     */
    public static int getLineNumber() {
      return Thread.currentThread().getStackTrace()[2].getLineNumber();
    }

      //Function to load the properties file
      private static Properties loadProperties(String fileName)
      {
            InputStream propsFile;
        Properties tempProp = new Properties();

        try
        {
            propsFile = new FileInputStream(fileName);
            tempProp.load(propsFile);
            propsFile.close();
        }catch (IOException ioe) {
            System.out.println("I/O Exception.");
            ioe.printStackTrace();
            System.exit(0);
        }

        return tempProp;

    }

}
```

Appendix 4 (Plugin.xml's source code)

```
/*
*  plugin.xml source code
*/

<?xml version="1.0" encoding="UTF-8"?>
<?eclipse version="3.4"?>
<plugin>

   <extension
         point="org.eclipse.ui.views">
      <category
            name="PCSE Plugins"
            id="org.example.TableViewer">
      </category>
      <view
            name="Timelog"
            icon="icons/sample.gif"
            category="org.example.TableViewer"
            class="org.example.tableviewer.views.TimelogView"
            id="org.example.tableviewer.views.Timelog">
      </view>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.jdt.ui.JavaPerspective">
         <view
               ratio="0.5"
               relative="org.eclipse.ui.views.TaskList"
               relationship="right"
               id="org.example.tableviewer.views.Timelog">
         </view>
      </perspectiveExtension>
   </extension>
   <extension
         point="org.eclipse.help.contexts">
      <contexts
            file="contexts.xml">
      </contexts>
   </extension>


   <extension
         point="org.eclipse.ui.views">
      <category
            name="PCSE Plugins"
            id="org.example.TableViewer">
      </category>
      <view
            name="Project Dashboard"
            icon="icons/sample.gif"
            category="org.example.TableViewer"
            class="org.example.tableviewer.views.ProjectDashboardView"
            id="org.example.tableviewer.views.ProjectDash">
      </view>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.jdt.ui.JavaPerspective">
         <view
               ratio="0.5"
               relative="org.eclipse.ui.views.TaskList"
               relationship="right"
               id="org.example.tableviewer.views.ProjectDash">
         </view>
      </perspectiveExtension>
   </extension>
   <extension
         point="org.eclipse.help.contexts">
      <contexts
            file="contexts.xml">
      </contexts>
   </extension>

   <extension
         point="org.eclipse.ui.views">
      <category
            name="PCSE Plugins"
            id="org.example.TableViewer">
      </category>
      <view
            name="Size Matrix"
            icon="icons/sample.gif"
            category="org.example.TableViewer"
            class="org.example.tableviewer.views.SizeMatrixView"
            id="org.example.tableviewer.views.SizeMatrix">
      </view>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.jdt.ui.JavaPerspective">
         <view
               ratio="0.5"
               relative="org.eclipse.ui.views.TaskList"
               relationship="right"
               id="org.example.tableviewer.views.SizeMatrix">
         </view>
      </perspectiveExtension>
   </extension>
   <extension
         point="org.eclipse.help.contexts">
      <contexts
```

```
            file="contexts.xml">
      </contexts>
   </extension>

   <extension
         point="org.eclipse.ui.views">
      <category
            name="PCSE Plugins"
            id="org.example.TableViewer">
      </category>
      <view
            name="Changelog"
            icon="icons/sample.gif"
            category="org.example.TableViewer"
            class="org.example.tableviewer.views.ChangelogView"
            id="org.example.tableviewer.views.Changelog">
      </view>
   </extension>
   <extension
         point="org.eclipse.ui.perspectiveExtensions">
      <perspectiveExtension
            targetID="org.eclipse.jdt.ui.JavaPerspective">
         <view
               ratio="0.5"
               relative="org.eclipse.ui.views.TaskList"
               relationship="right"
               id="org.example.tableviewer.views.Changelog">
         </view>
      </perspectiveExtension>
   </extension>
   <extension
         point="org.eclipse.help.contexts">
      <contexts
            file="contexts.xml">
      </contexts>
   </extension>

</plugin>
```