SENIOR DESIGN GROUP 13

Project Plan

Ptolemy Eclipse Plugin

Version 1.3

Kimberly Eue, Adam Rotondi, Justin Landsgard, Tim Flanigan , Kyle Bradwell

4/9/2012

This document encompasses what factors we have considered in the creation and management of this project involving developing a plugin for the Eclipse IDE, for use with the Ptolemy programming language. For technical and implementation factors, please see the Design Document.

1: Introduction

This section covers general information required to view the rest of this document, including those involved in this document and the goal of this document.

1.1: Table of revisions

Revision number	Date of revision	Summary of changes
1.0	3/14/2012	Initial project plan document
1.1	3/22/2012	Restructuring of document, added numbered sections.
1.2	4/1/2012	Corrections made as per client review, more consistent mapping to project plan.
1.3	4/9/2012	Added more tables to indicate versions and mapping between requirements and iterations. Corrected syntactic errors.
1.4	10/6/2012	Added and corrected language in several sections to be clearer on intent. Added several new terminology items to reflect this.

1.2: Team Members

The following are team members in this project, listed with their roles. All team members are equally assigned to development roles as their skills allow.

Team Member	Role
Adam Rotondi	Team Leader
Kyle Bradwell	Communicator
Kim Eue	Web Developer
Tim Flanigan	Communicator
Justin Landsgard	Web Developer

1.3: Goal of this document

The intended purpose of this document is to cover the description, acknowledgement, and breakdown of work, resources, and scheduling across the length of this project. This document is not intended to describe on a technical level the processes used in designing or creating the end results desired by this project.

1.4: Table of Deliverables

Deliverable	Delivery Date	Prototype	Completed
Project Plan v1.0	Feb 14, 2012	None	Feb 14,2012
Iteration 1 Design Document v1.0	Feb 28, 2012	Small Eclipse Plugin Example	Document Mar 1, 2012. Prototype Delivered Mar 8 2012.
Project Plan v1.1	Feb 28, 2012	None	Feb 24,2012
Iteration 2 Design Document v1.0	April 16, 2012	Example Syntax Highlighting in eclipse	Document September 15, 2012 Prototype October 22, 2012
Iteration 3 Design Document v1.0	September 2, 2012	Example altered Stack Trace in Eclipse	n/a
Final Integrated Project Plan and Design Document	October 1, 2012	Final Deliverable prototype	December 3, 2012

1.5: Table of Terminology

Term	Description
IDE	A set of tools for programming, often composed of a text editor, a compiler, and several advanced features that allow a programmer to easily develop code in a specific language.
Eclipse	A popular IDE for Java, with extensions to other languages via “plugins”
Editor	In the Eclipse IDE, an Editor is a window or component where code or text can be entered. An editor can have properties to make it more suitable for software development than a standard text editor such as Notepad.
Event	In this document, the single term Event will be used to reference a new feature in the Ptolemy language, on the same level as classes and interfaces. This new type has its own features that we cover in the document in 3.2.1
Keyword	Within Eclipse and many other IDE’s, a Keyword is a string specially interpreted by the compiler or other sources, such that its meaning is very context sensitive. Developers often like IDE’s to represent keywords specially to reflect this.
Outline View	A part of the Eclipse IDE that allows a user to view visually in an organized manner all variables, methods, classes, and interfaces inside of a given .java file.
Perspective	Within the Eclipse IDE, a perspective represents a set of tools or settings specific to a desired functionality. Perspectives may have overlapping functionality.
Project	Within the Eclipse IDE, a project is a collection of source code and other files to be used as part of a software product.
Plugin	In this document, Plugin refers to an extension of the Eclipse IDE that either overrides previous features or provides new features (or both).
Ptolemy	A new programming language derived from Java, with new features. See 3.2.1
Package Explorer	A part of the Eclipse IDE that allows a user to determine the hierarchy of classes, packages, and individual files inside of a Java project.
Refactoring	The processes of modifying source code in a complete and automated way. Examples of Refactoring include:

	• renaming a variable in all areas of a file • Extracting a section of code into a separate method, and calling that method instead of rewriting the code
Stack Trace	Within computer science, the stack is a data structure used by the underlying OS to correctly execute programs. A stack trace is the log of the execution of items on the stack at a certain time. Stack traces are often crucial for developers to determine their programs are working correctly.
Wizard	A feature in Eclipse that helps the user set up files such as Classes, Interfaces, etc. in a standard way through the use of pop up windows with input for parameters. After receiving input, the class/interface is generated with those parameters specified.

2: Overall Description

This area covers the general description of the problem as well as the team's goals in solving the problem.

2.1: Problem Statement

As the field of computer science continues to grow and evolve, new problems that arise in the field are being approached. In both industry and research, new solutions are explored to adapt to these problems, one being the creation of a new programming language.

This is the case with Ptolemy, an extension of the Java programming language, which supports additional features that make it attractive to solve problems faced by larger programs and event driven programming.

Currently, the Ptolemy language does not have an easily distributable method of use that many programmers or students would have access to. This inhibits the usability of the language to solve many industry problems.

2.2: Goal Statement

Our goal as a team is to create a development environment suitable to be used in the creation of Ptolemy programs, for the popular Java IDE, Eclipse. This plugin will boast features suitable to develop in the Ptolemy language. This plugin will be able to boast creation and development of industry level programs in the Ptolemy language.

3: Proposed Work

This section covers the individual tasks proposed by the client to be developed by the team. Technical details will be as little as possible to not overlap with the design document.

3.1: Visual Changes in Eclipse

These changes are functional requirements as well as look and feel requirements for this project. They help further the use of Ptolemy in eclipse as described by each member item. While each of these

requirements may require further changes to other aspects of Eclipse, the purpose of these items is to represent the Visual changes that will be the final results.

3.1.1: Perspective creation for Ptolemy

The Ptolemy language is an extension of the Java language, but must be handled by a separate compiler. To easily distinguish when this should occur, the team will develop an Eclipse **perspective** to easily distinguish the complete set of changes to Eclipse involved. This also helps to emphasize that changes in this perspective only apply while in this perspective, and do not change normal functionality of the "standard" java perspective in eclipse.

3.1.2: Modify menu and buttons

Several of the menu actions and buttons on the standard eclipse toolbar must be modified to handle the functionality of Ptolemy, and to distinguish a Ptolemy **project** from a Java **project**. The members below fall into this category with their own specifics. Menu options not listed for change should remain unchanged from their previous functionality. This emphasizes Java functionality is part of Ptolemy, and those areas needing no change should be left alone.

3.1.2.1: New Ptolemy project

There will be an option in the menu under File->New to make a new Ptolemy **project**, in a similar manner to the creation of a Java **project**, using **wizards**. The other features of the file system and menus should remain unchanged.
See Figure 1 for an example of what the user will see when trying to create a new Ptolemy project in Eclipse.

3.1.2.1: New Event

A new type of file can be made using Ptolemy, called an **Event**. This type should have a similar creation process as classes, interfaces, and packages in Eclipse. The creation should use similar **wizards** for its creation as classes, packages, and so on. Other features such as class creation, Interface creation, etc. should remain unchanged.
See Figure 1 for an example of what the user will see when trying to create a new Event in Eclipse.

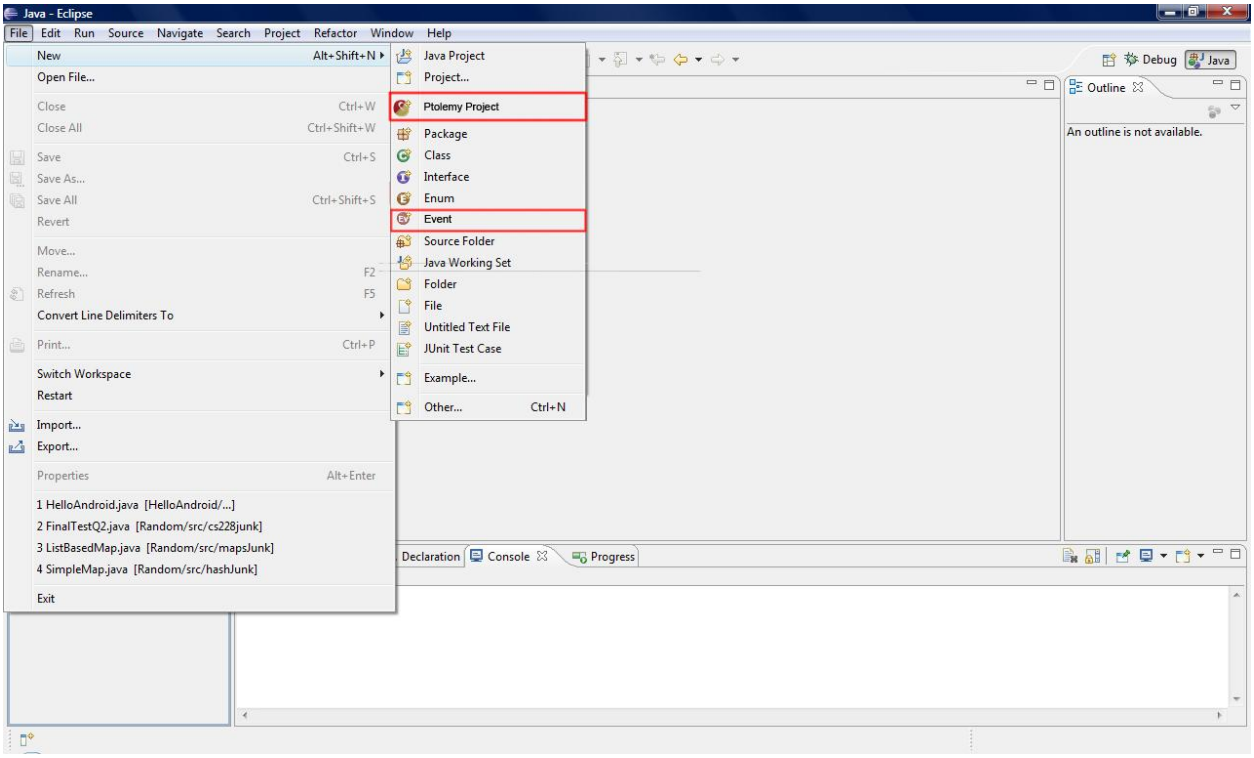

Figure 1: Creating new Ptolemy Items

3.1.3: Package explorer

The **Event** type must be accurately represented in the organization of the package explorer. To do this, the package explorer will use a different symbol for Events as opposed to Classes, Interfaces, etc. Other standard conventions in the Package explorer will remain unchanged.

Additionally, the **Event** type is saved in standard ".java" files, so the files containing **Events** will show up as ".java" files, in keeping with current features of eclipse. This will help those developers already acquainted with java and Eclipse adapt to using Ptolemy quickly and easily.

See Figure 2 for an example of how Ptolemy files would be represented in Eclipse.

3.1.4: Outline view

The outline view of a **project** will also need to represent **Events**, in a similar fashion to the package explorer, previous functionality will remain unchanged while adding the feature of displaying **Events** as their own unique type. This will help developers who are already familiar with Eclipse's outline view use it for Ptolemy development.

See Figure 2 for an example of how **Events** or the other new types would be represented in Eclipse.

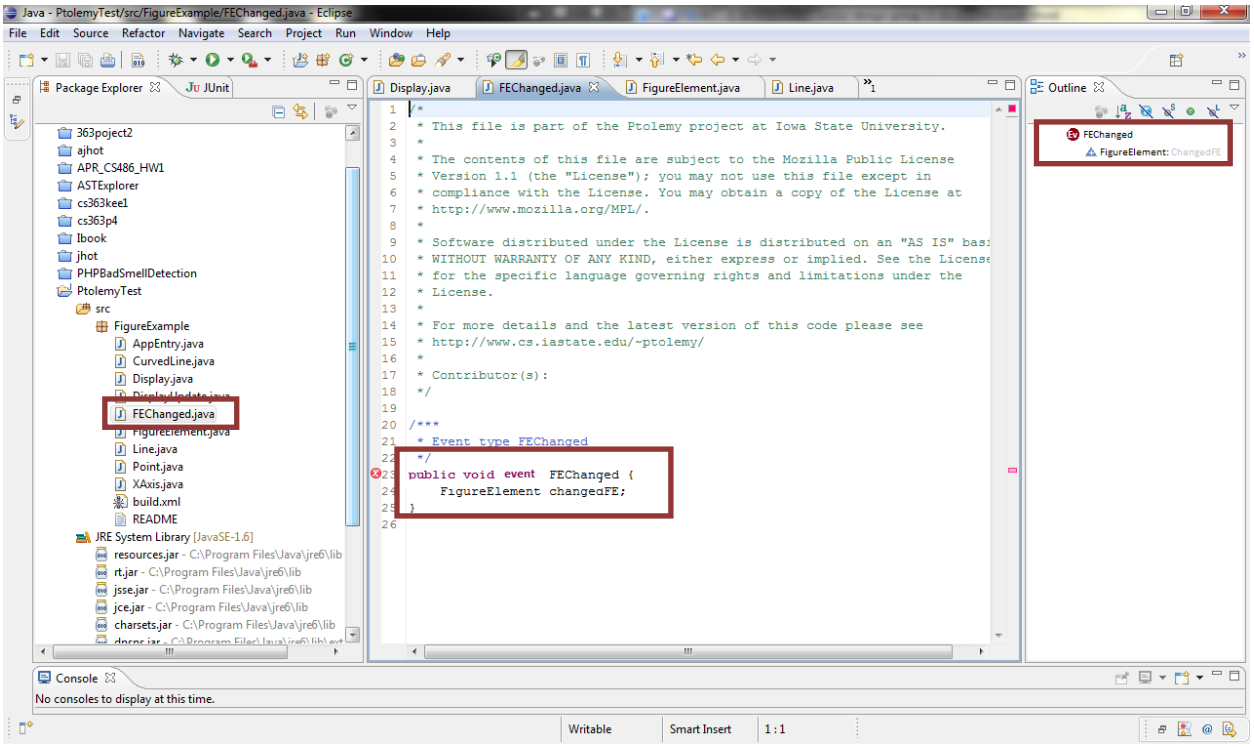

Figure 2: Sample Syntax Highlighting and Event notation

3.2: Behind the Scenes Changes

These changes are also functional requirements of this project, but deal with less visual changes to the Eclipse IDE. These changes however could influence the display of code or other requirements of this project in Eclipse, but their focus is primarily outside of the scope of purely visual changes. Again, technical implementation details will be avoided as to not overlap with the Design Document.

3.2.1: New keyword support

Some keywords are introduced in Ptolemy that is not supported by the Java language. These keywords break down into 2 major categories, described below.

3.2.1.1: New General Keywords

There are a few new words that when typed in eclipse's **editor** should be represented within Eclipse's code as **Keywords** for the purposes of the compiler and Eclipse's display of these words in the **editor** (see *Syntax Highlighting*). Those **Keyword**s are:

- event
- announce
- when
- register()

NOTE: it has been said to us by our client that in the future, the statements register() and invoke() would like to have their parenthesis removed so that they do not appear to be a method call, and better match the other keywords. However, modifying the compiler to do so is beyond the scope of this class.

3.2.1.2: Context Sensitive Keywords

Ptolemy introduces a number of **keywords** that would only need to be represented as such when viewed from a specific context (e.g. inside of a specific block of code), and they should be represented as such.

Those words are listed below, as well as their context in which they should be represented as **keywords** for the purposes of the compiler and Eclipse's display (see *Syntax Highlighting*). If not within their keyword context, they will be represented as regular words if possible.

- assumes - Inside of an event or its handler.
- ensures – Inside of an event.
- establishes – Inside of an event or its handler, but if inside the handler must be preempted by **refining.**
- invoke() – Inside of an event handler or event declaration.
- old – Inside of an Event or its handler.
- refining – Inside the handler of an event, and only valid in front of **establishes**.
- requires - Inside of an Event or its handler.

3.2.2: Code completion

The functionality for Eclipse to complete sections of code for java projects should be implemented for use in Ptolemy. As such, when a user types a method call or uses a "." for objects, the plugin will display a list of applicable variables or methods respectively that can be used. The functionality of java's code completion for java constructs will remain unchanged, while support for Ptolemy constructs will be added. This again is to help developers already familiar with java and it's practices in eclipse use the Ptolemy language easily.

3.2.3: Syntax Highlighting

Ptolemy introduces a new set of **keyword**s to the rules of the Java language, distinguishing it from Java's syntax. These new keywords, however, must visually match those already supported by Java, so that they are easily interpretable by the user as a system keyword, when appropriate.

Previous keywords in java should remain unchanged.

See Figure 2.

3.2.4: Refactoring support

Just as the Eclipse IDE supports functions such as **refactoring** for objects, the Ptolemy plugin will also provide the same functionality for the new items included in the plugin. Previous **refactoring** features and code for java and those areas of Ptolemy identical to java should remain unchanged.

3.2.5: Debugging Support

The team will implement debugging features to help developers using Ptolemy to correctly interpret their program's **stack trace**.

Currently, the compiler for Ptolemy during a standard method call under the new **Event** type will expose additional internal implementation elements of Ptolemy on the **stack trace** that should not be viewed by Ptolemy developers. While this may have been useful in the past to identify the Ptolemy compiler's methodology, it is not useful in an industrial setting, and should be hidden to show only "client" code. The ability to debug normally in Java and those areas of Ptolemy identical to java should remain unchanged.

4: Estimated Resources and Schedules

This section covers the Resource allocation of the project, including man hours and monetary expenses, scheduling of man hours, and budgeting of expenses.

4.1: General Project Schedule

This project will follow an iterative design schedule, to better allow for changes in the design, eliciting requirements, and to follow standard industry practices for software development.

4.1.1: Table of iteration requirements

This table is meant to show the requirements that will be implemented during each iteration of this project. For technical details of the iterations, please see the design document.

For non technical details of the iterations, see the sections below.

Iteration number Requirement	Requirement Number	1	2	3
Perspective Creation for use with Ptolemy	3.1.1	X		
Modify menu and toolbar items for use with Ptolemy	3.1.2	X		
Modified Package Explorer view for use with Ptolemy	3.1.3	X		
Modified Outline view for use with Ptolemy	3.1.4		X	
Support for new keywords used in Ptolemy	3.2.1		X	
Code completion for new Ptolemy keywords	3.2.2		X	
Syntax highlighting for new keywords used in Ptolemy	3.2.3		X	
Refactoring support for new Ptolemy keyword items	3.2.4			X
Debugging support for Ptolemy programs	3.2.5			X

4.1.1: Iteration 1

The first iteration will deal with the most visual elements of the Ptolemy plugin. Namely developing the modified look of the interface, developing the perspective, and the tools involved.

Requirements to be met:

- 3.1.1 : Perspective creation for use with Ptolemy
- 3.1.2 : Modified package explorer view for use with Ptolemy
- 3.1.3 : Modified outline view for use with Ptolemy

Testing the correctness of this iteration will be fairly simple. If it meets the requirements and passes usability testing for our client to use, we shall consider it a success.

4.1.2: Iteration 2

The second iteration will delve into deeper complexity, and focus on implementing Syntax Highlighting, keyword support, and running Ptolemy Programs.
Requirements to be met:

- 3.1.4 : Modified Outline view for use with Ptolemy
- 3.2.1 : Support for new keywords used in Ptolemy
- 3.2.3 : Highlighting of new keywords used in Ptolemy files
- 3.2.2: Code completion for new Ptolemy keywords.

Testing this iteration's modules for correctness is more difficult than iteration 1, and test cases will need to be made to ensure that most if not all possibilities or equivalence cases are covered with respect to Ptolemy syntax combinations. Additionally, tests will need to ensure that Ptolemy context sensitivity with some keywords are in place, and that Ptolemy differences are well integrated with java code.

4.1.3: Iteration 3

The third iteration will deal most notably with the Refactoring and debugging support, and will be the most complex iteration.
Requirements to be met:

- 3.2.4: Refactoring support for new Ptolemy keyword items.
- 3.2.5: Debugging support for Ptolemy programs.

Testing this iteration will be the most difficult of the three iterations, and a suite of test cases will be developed to ensure that these modules are reasonably covered.

4.2: Overall timeline

This project will be completed over the course of 2 semesters (approx. 8 months). Starting in January of 2012 and ending in December of the same year. The project will follow the schedule described above. For each iteration, the expected dates for start and completion are as follows:

Iteration number	Expected Start date	Expected End date	Actual Start date	Actual End date
1	March 1 2012	April 15 2012	March 1 2012	April 30 2012
2	April 16 2012	August 20 2012	Aug 20 2012	December 3, 2012
3	August 21 2012	December 10 2012	TBD	TBD

4.3: Expenses

There are no expenses at this time other than the man hours involved in development.

Additional Information

This section covers reference or non-functionally relevant information dealing with this project.

Reference Information

Client –ISU Computer Science Dept.

Advisor –Hridesh Rajan, hridesh@iastate.edu

Additional members involved in the project

Simanta Mitra: Faculty – smitra@iastate.edu

Robert Dyer: PHD Student – rdyer@iastate.edu

Contact Information

Team 13 email list – dec1213@iastate.edu

Website – http://seniord.ece.iastate.edu/dec1213/