COMP 4905 Honours Project Report

One-click Java debugging tool for Eclipse

Author: Jiaqi Zhu (100653388)

Supervisor: Dwight Deugo

School of Computer Science

Carleton University

2009.12.09

ABSTRACT

Testing plays a significant role in the life cycle of developing software products. For Java developers, many debugging tools are available since open-source software become quite common. Built upon the concept of plugin, Eclipse is the powerful toolbox loads with testing tools. However, there lacks a manager of the tools which could run all the tests on the software under development and generates user-friendly test reports in one-click. To introduce such a tool, we find a solution by using CruiseControl, an automatic integration tool, along with Eclipse and three in-famous testing/debugging tools: JUnit, Findbugs and PMD. In this project, an Eclipse plugin in created to act as an interface between CruiseControl and user of Eclipse. User is able to use the one-click testing functionality provided by CruiseControl and the debugging tools. This project also provides easier configuration of CruiseControl.

This report will explain the design and implementation of this solution and validate the improvement of efficiency when applying this solution to a programmer's work.

ACKNOWLEGMENTS

I would like to express my deepest gratitude to all those who helped me during the working of this honour's project, especially to my supervisor Professor Dwight Deugo who gave me considerable help by means of suggestion, comments and criticism on the project and this writing report.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 OVERVIEW

Traditionally a software developer has to build his project manually for testing purpose. The test results are usually outputted to several different places and therefore they may not be well organized. It is sometimes not a big problem for individual programmer when the total size of the projects he works on is relatively small. However, when the projects are large-scaled and the amount of projects being developed at the same time is great, running all tests manually could be very time consuming. It is even more problematic if the test plan for all projects is not well designed or the works are not organized. As a solution, using project manager software, which also does automatic building and testing for software projects as well as organizing the works, would have saved software developers' time.

CruiseControl is an automatic integration tool for Java developers as a project manager. It controls how and when the projects should be built and tested. It also provides support to several testing tools which could be used later, for example, JUnit, Findbugs, and PMD. With the support of Http and RMI technology the results could be merged and published to a web server from where the programmers can see the results in an organized way.

CruiseControl saves Java programmers' time. However, in order to take advantages of CruiseControl, they must learn to use the software. Sometimes they may find it difficult because configuring CruiseControl is not an easy task. A developer has to have knowledge beyond Java in order to adopt the software. For example, the main configuration file of Cruise is written in XML. Thus who wants to edit the file has to know the XML language. The developer also has to know how to use a batch builder, for example, Ant or Maven. Moreover, in order to set up a web server which collects all the generated reports, he has to know about how to write servlets and how to use testing tools which he wishes to run on his projects. Last but not least, in order to make everything works with each other, he also has to change several files located in the local file system or on a remote machine.

It is expected that CruiseControl will improve the efficiency of developing software projects. However, the programmers will hesitate if they have to learn a lot about using the software before they can actually use it. Therefore, they would hope that there exists a ready-out-of-box solution which has all the functionalities and is easy to learn and use.

CruiseControl has the potential to be such an ideal tool as long as there is an easier way to configure it before application. It will also be helpful if

CruiseControl's functionality is integrated in a widely used software developing environment therefore the works could be done in-place.

For many software developers, Eclipse is the software developing environment for Java. To reduce the difficulty of configuring CruiseControl, we can provide a plug-in with functionality to configure CruiseControl. The plug-in will be able to provide great help to developers because all the works could be done in-place thereafter.

1.2 GOALS

The goal of this project is to provide a one-click debugging tool for Java developers who use Eclipse, in order to improve their work efficiency of developing software projects. This debugging tool should make use of CruiseControl's power as a project manager and an automatic integration tool, so that the tests could be done automatically while the programmers are working on codes at the same time. The debugging toolbox will contain JUnit, Findbugs, and PMD. The test results of running the tools should be available immediately after each test and should be presented to the programmers in an organized way, therefore they can fix the bugs quickly. A programmer should be able to configure CruiseControl within Eclipse and it should be designed easy enough for someone new to the software to start.

1.3 OBJECTIVES

The best way to achieve the goals is to design an Eclipse plug-in that works as a connection between a programmer and CruiseControl. The programmer is able to use the functions provided by the interface to manage his projects under CruiseControl in the following ways:

1. The plug-in should provide shortcuts to configuration files for CruiseControl to build and test the project automatically.
2. The plug-in should provide shortcuts to a project's build file for a programmer to define the rules for testing and building the project, so that the rules could be used by CruiseControl. Several testing tools should be available to the programmer.
3. The plug-in should provide shortcuts to the build results and test results. The results should be published to a web server and the programmers should be able to analyze them.
4. The plug-in should provide shortcuts to run and shutdown CruiseControl so the changes made to CruiseControl and projects could be applied.
5. The plug-in should provide shortcuts to the documentation of CruiseControl.

It is complicated to add new testing tools to CruiseControl for people who are not familiar with the software. Therefore three popular testing/debugging tools are selected and they should be pre-configured in CruiseControl. The tools are JUnit, Findbugs and PMD. They should provide sufficient testing for most of the projects.

For the similar reason, Jetty is chosen as the servlet container and should be pre-configured with CruiseControl. The generated reports should be submitted to a Jetty server and developers should be able to check these reports via graphic interface provided by servlets.

For this project it is assumed that the Ant builder is used to build the software projects because it is supported by Eclipse and used by many Java developers. Therefore CruiseControl should be pre-configured to use Ant.

There will be three stages for developing this project:

1. Integration of the testing and the debugging tools with a binary distribution of CruiseControl.
2. Development of the Eclipse plug-in as a front-end CruiseControl configuration program.

3. Deployment of Jetty with the functionality of handling reports generated by JUnit, Findbugs and PMD.

It is expected that the when all the objectives are met, there will be an Eclipse plug-in which provides a one-click debugging solution to the Java programmers.

1.3 REPORT OUTLINE

The remainder of this report is organized as following: Section 2 is the background knowledge of all the software used to complete this project. In Section 3 there is an in-detail explanation of the design and the implementation of this project along with a list of assumptions made. In section 4 there will explain how well this project meets the objectives as stated in Section 1. There is also a comparison between this project and an existing configuration tool. Section 5 will be the conclusion with a discussion on possible future works for development of this project.

2. BACKGROUND

Of the many software projects used to complete this project, the most important three are CruiseControl, Eclipse, and Apache Ant, which constructed the framework of this one-click debugging tool. There are several other software and technologies used in completing this project, they will be introduced briefly afterward.

CruiseControl

CruiseControl is both a continuous integration tool and a framework for making a customized continuous build process. It is written in Java and is supported by many builders such as Ant and Maven. [1]

There are three main components of CruiseControl:

The most important one is the build loop. It is configured in a XML file so all the projects managed by the software can be built following a schedule. It has a monitor for events so that the build process can be triggered upon certain changes.

The second part is the JSP reporting application. By using Http and RMI technologies, reports generated by the build loop can be published to a server. Users of CruiseControl can write servlet applications to handle the reports and the artifacts of projects.

The last part is a graphic user interface that shows all projects' status under the build loop. [2]

The following is a diagram showing the architecture of CruiseControl. [2]

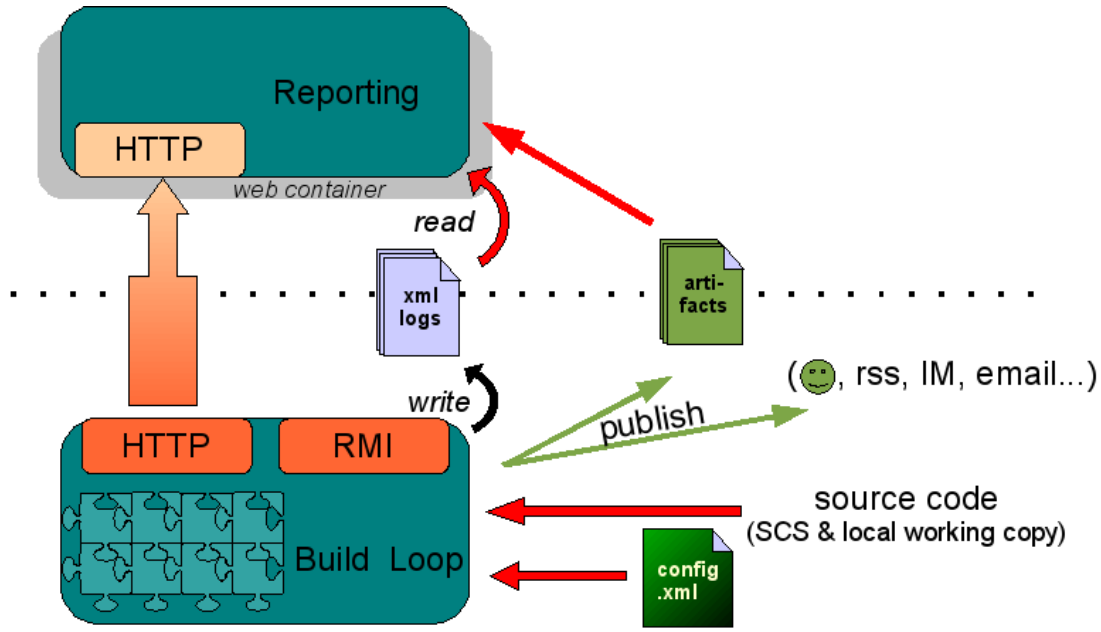

Figure 1: [2] Architecture of CruiseControl

A typical build cycle in CruiseControl consists of several parts: The software first determines whether a build is necessary. If a build needs to be done for a project, the project's builder file will be acquired and built by CruiseControl.

Log files will be generated with any custom information with respect to the project and the status of the build process itself. These logs will be then merged and reported using the JSP reporting tool. [3]

Eclipse

Eclipse is a software development environment comprising an IDE and a plug-in system. It has a small run-time core and the functionally is provided by the plug-ins. The plug-ins are portable and could be deployed as user's requirement to provide all kinds of different features such as supporting of programming languages, graphic interfaces, debugging tool, etc. [4]

Built upon OSGi framework, Eclipse runtime (Equinox) could be extended dynamically with bundles that are cohesive, self-contained units which explicitly define its dependencies to other modules / services and explicitly defines its external API. The bundles are .jar files with additional meta-information which is used by OSGi runtime to load and to unload the functions provided within the bundles dynamically. Eclipse extends the concept of bundles with extension points. The architecture of a smallish core runtime plus plugins which provide customized functions made Eclipse a powerful yet flexible IDE [5]

Apache Ant

Ant is Java-based build tool which works similar to Make. The main concept Ant is built on is for multi-platform. In order to achieve that, Ant is written in Java and its scripts are written in XML. [Ant.Apache.org, 2009] Ant is even more powerful with the extensibility. Developers can extend Ant by writing Java classes and thus develop custom Ant Tasks. For example, to enable JUnit in Ant one could define <junit> tag and use it in the content of an Ant script where a JUnit test is required. [6]

Ant uses a XML file to store the build information of a software project. The tags form a list of tasks to be done. A project's build file always starts from a <project> tag, usually followed by several <target> tags which defines a series of tasks. An Ant script can also have several <property> tags which contain name-value pairs that are accessible from the other tags. Custom ant tasks are defined after <project> and before any <target>. Ant tasks are used in contents between a fair of <target> </target> tags. [6]

Here is a diagram shows the structure of a basic Ant script:

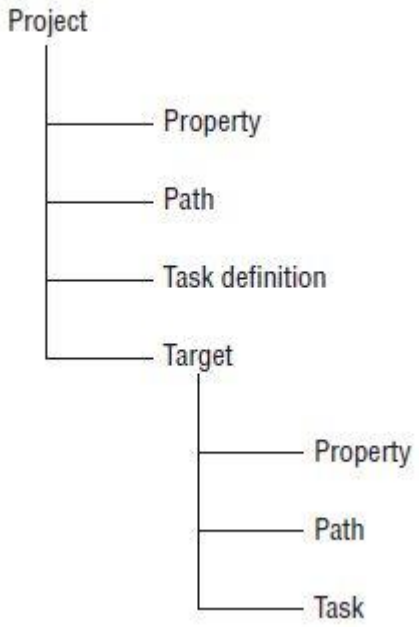

Figure 2: [7] Structure of a basic Ant Script

Jetty

Jetty is an open source Java-based Http server and servlet container. It is developed to focus on providing a simple, efficient web server and could be embedded in Java programs to provide web services [8].

Unlike Tomcat which runs as a stand-alone application server, Jetty is designed small and could be easily embedded into an application or a framework to provide network functionality. [9]

JUnit

JUnit is an instance of xUnit architecture for unit testing framework. As the name suggested, JUnit is for Java programming language and it is the standard testing library for Java. The framework is simple and the tests written in this framework is repeatable. It is important to test-driven software development approach. [10]

The major improvement from JUnit 3 to JUnit 4 is the later version takes advantage of Java 5's annotation feature. The syntax become clearer and using of the tool becomes much simpler. [11]

Findbugs

Findbugs is a software project for detecting bugs in Java code. It uses static analysis to identify potential errors. It operates in Java byte code rather than the source code. Findbugs is distributed as a stand-alone program and an Eclipse plug-in. [12]

For this project the stand-alone distribution is used.

PMD

PMD is a static code analyzer based on static rule set. It scans Java source code to identify problems. Not only pinpoint actual errors in the code, it also notices programmers for inefficient codes. It is a popular debugging tool and is integrated in Eclipse and Ant. [12]

CVS

CVS is a version control system. Users of it can keep record of the library of source files, documents and other files used in a software project. Therefore several software developers could cooperate in same project. CVS uses a server /client based architecture. The complete copy of a software project is stored on a server. A programmer could retrieve the latest copy of the project using a client application. Changes made on a local copy could be committed to the server later. [13]

Servlets and JSP

Serlvets are server-side Java code written to dynamically generate contents or to perform actions. All servlets implements the Servlet Interface which defines a life cycle of the code. The servlets are used to provide web-based services.

They run on a servlet container, for example Jetty. The servlets overrides the service() method defined in the interface and the method is called to handle client requests. [13]

A JSP is a Java server-side page that will dynamically generate contents in response to client side requests. A JSP page consists of texts written in a mark-up language, for example Html or XML, and scriptlet tags which make use of the JSP API. The JSP page is later compiled to servlets and the client requests are handled. [13]

XSLT

To generate an HTML file from XML, an XSLT style sheet file is created. An XSLT style sheet contains the instructions for the transformation a programmer wants to perform on an XML file. XSLT style sheets are built on templates which specify what to look for in a XML source input and what to output to the result file. The input files and output files could be written in any tree-based markup language. Therefore the XSLT technology could also be applied to convert a XML file to another XML file. [14]

All the software projects listed above are open source software. They are used in this project. The software could be downloaded freely on internet from their

developers' websites.

3. APPROACH

3.1 ASSUMPTIONS

Before starting work on the Eclipse plug-in project, several assumptions are made. These assumptions are based on the objectives as claimed in the introduction chapter and are solely for the purpose to assure users that the plug-in will make it easy to use CruiseControl.

First, the plug-in is not responsible for setting up CVS server. It is assumed that the CVS server is properly set up and managed. Since CruiseControl implements functionality of a CVS client, it is not necessary for the plug-in to provide functionality to configure, start, restart, and shutdown a CVS server.

Second, users of this project application are not encouraged to change existing functionality of CruiseControl. A user may change the functionality if he wishes to do so. However, this project is not designed for that purpose thus no helps will be provided.

Third, if a configuration file is to be edited by a users, the plug-in will not provide checks for syntactic and semantic correctness, thus if there is an error resulted by changing the files which would cause a failure of CruiseControl or the other software, that error is not going to be caught before it occurs. It is user's responsibility to make sure the configuration files are correct.

Fourth, the Eclipse plug-in will not do any search for any of the configuration files. User is required to provide correct information about the paths to the files. Additionally, whenever user is prompt for a file, the default path is at the root of the file system. It is obvious that user will find it easier if a file chooser is opened on directory which contains the file rather than going from the root, however sometimes the build files are kept in several locations. For this reason, it is unnecessary to assume one of the files is going to be used and set it to be default. It is also a potentially dangerous action to assume the location of a key file because it may crash the software.

Fifth, since having multiple things which do the similar job is confusing, shortcut to the Dashboard component is not provided. There will be only one html based user interface provided to users. Users may use Dashboard from the external if desired. However it will not be extended to support the new debugging tools.

Last but not the least, this project is developed for Windows platform (Vista and later). Although it is more useful if it supports multi-platform, more works have to be done and the final product will be significantly larger because different software sets are included to support different platforms. At this moment it is more important to find a solution to the problem than applying the untested solution to multi-platform. Therefore it is a personal preference to start developing this project with a platform that I am familiar with.

3.2 DESIGN OF THE PROJECT

The Eclipse plug-in of this project will keep records of all the file locations and URLs to web based services in a XML based configuration file. A template is provided to users in creation of such a file so the information is collected and could be used later by the plug-in. The required information includes: where CruiseControl is installed, what the address for the server that reports are sent to is, the ports used, and where the project's build file is located. When using the software, the user activates a configuration and the information will be used to create the shortcuts.

A programmer can create multiple files for the same project under development for different setting of the software. To switch from one setting to another the user is only required to activate a different file.

To keep all the works inside Eclipse so that a user does not need to switch between programs when developing a software project, Eclipse's internal editors and web browser are used. When a configuration file is to be edited, it is opened in an Eclipse editor on the currently working bench. The same editor is used if the target files are same. Eclipse's internal web browser is used to open URL's to CruiseControl's User interface and to issue commands to shutdown or restart CruiseControl. (Requires knowledge about port number of CruiseControl's JMX console)

The following defines the typical using of this software and is considered in designing this plugin project:

1. User activates a configuration file for the plug-in.
2. Run CruiseControl.
3. User can open CruiseControl's user interface to see the previous build reports or trigger a new build.
4. User edits the configuration files to change the build/test rules.
5. Restart CruiseControl to apply the changes.
6. Go to the UI and build the project and new reports are generated.
7. Keep developing or shutdown CruiseControl.

To use this plug-in as a one-click debugging tool, a user defines the tests in a project's build file. After running CruiseControl, the software will test the project under its management with pre-defined debugging tools automatically, and the results will be reported to the server. To improve the coverage of finding bugs, Findbugs and PMD supports are added to CruiseControl which already has JUnit 4 in its library. Adding Findbugs and PMD to CruiseControl requires knowledge beyond Eclipse, Ant and XML, making it a difficult work for people who are new to CruiseControl. To avoid this situation, it is suggested to include the debugging tools in the latest distribution of CruiseControl, pre-configure it and create a ready-to-use toolbox. To include the tests in a project, a user simply needs to use the pre-defined ant tasks and put the keywords in the project's build file.

3.3 DESIGN OF USER INTERFACE

To keep the Eclipse plug-in simple and easy to use, there will be no new view and no new perspective added to Eclipse. Instead, it shows a drop-down list of buttons when the user right-click on a Java project from Eclipse's resource view. In Eclipses there are two kinds of projects: Project and Java Project. Since the Eclipse plug-in is designed to help Java developers. It will only show the drop-down menu on Java projects.

On the drop down list there are buttons that provide access to the configuration files that are involved in adding, modifying, and removing a project under CruiseControl's management: CuriseControl's config.xml file and the project's Ant build file. Upon selection, editors are opened for the file within Eclipse and the user can freely make changes to the content.

Following on the list there are buttons to CruiseControl's html-based interface. The interface is a JSP running on the Jetty server. From the interface a user may access the reports generated after each build and test of the projects under CruiseControl's management. A manual build can also be triggered by clicking a button on that user interface.

Buttons for start/restart and shutdown CruiseControl are designed to apply changes on the list. Restart and shutdown buttons make use of CruiseControl's JMX Console. When they are clicked, a command is issued to the console to shutdown the system. The shutdown hook takes 1-3 seconds to turn CruiseControl off completely. Therefore when a shutdown command is sent, there will be a 3 seconds waiting before next operation. There could be multiple instances of CruiseControl running on one system and the shutdown hook will only work on an instance ran by the Eclipse plug-in. It is suggested that users only use the provided buttons to start and stop CruiseControl, but one can always turn the software off by closing the Command shell.

Here is a screen shot of the Eclipse plug-in that shows the UI and CruiseControl's interface running in Eclipse's built-in web browser:

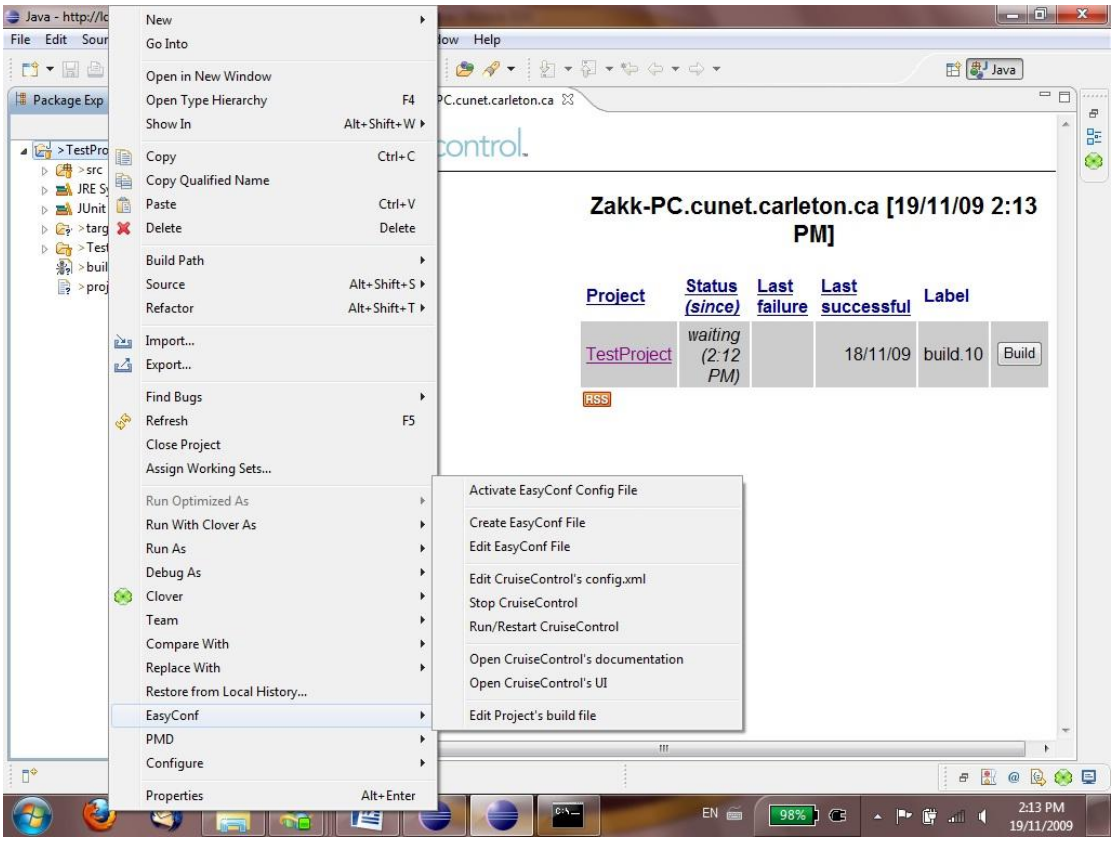

Figure 3: Interface of the Eclipse plug-in running on Windows 7

A configuration file of the Eclipse plug-in must be activated before the user can click on other items on the drop down menu. Otherwise error messages will pop up. If there are errors in the configuration file, the user will be warned with a "parsing error" message. If errors are discovered in the build file, corresponding error messages will be shown on the command shell running CruiseControl. The errors will also be outputted in build logs, which are accessible through CruiseControl's UI. The user interface will show status of

the builds in terms of success and failure.

3.4 IMPLEMENTATION

The CruiseControl configuration application in this project extends Eclipse's user interface by implementing Eclipse's "UI plugin" interface. The drop-down menu uses the popup menu extension of the IDE's UI. This plugin is integrated to Eclipse's UI and it activated when user right-click on a Java project in the resource explore view part. The activator class defines the life cycle of this plugin and is used by Eclipse to run the code along with plugin.xml file which defines extensions for this plugin.

Each button on the menu is linked to a popup menu action the corresponding .java file where code for that action is programmed. Besides of CreateConfig action, all other actions depends on the setPath(activate a config file) action where user is prompted for a file contains all the information this plugin needs to provide accesses to CruiseControl's services and configuration files. User must activate a configuration file for this plugin before he can execute any other action. After the file is chosen, the path to this file is stored in the Control class and an instance of XMLparser class is created in order to parse the required information about CruiseControl contained in the file. The data are then stored as name/value pairs in the XML Parser and could

be later used by the Control class.

When user clicks on a menu item, an action is triggered and it requires certain information about CruiseControl. For example, "run CruiseControl" action requires the port used by CruiseControl's control panel service. The Control object receives the request and gets the name/value pair contains required piece of information and returns it to the action, and thus that action is executed.

Executing CreateConfig action creates a configuration file of this plugin. A template is provided to users in creation of such a file so the information is collected and could be used later by the plug-in.

Here is a screen shot shows the template which will be provided to user:

```
<EasyConf>

    <!-- Path to the project's build.xml-->
    <Project>

    </Project>

    <!-- CruiseControl's server address -->
    <CCServer>

    </CCServer>

    <!-- CruiseControl's UI port number -->
    <CCPort>

    </CCPort>

    <!-- URL to CruiseControl's installation folder -->
    <CCInstall>

    </CCInstall>

    <!-- CruiseControl's JMX Console port number -->
    <CCConsole>

    </CCConsole>


</EasyConf>
```

Figure 4: Screen shot of the provided template

To explain the entries and their usages on the template, followed is a table shows the actions and the information needed to execute the action:

Action Name	Required Information (Name/Value as stored in XMLParser)	
	Name	**Value**
Edit CruiseControl's config.xml	CCInstall	Path to the where CruiseControl is installed
Edit the Project's build.xml	Project	Path to the build.xml file of the project

Start/Restart CruiseControl	CCConsole	Port number of where the JMX console is running
	CCInstall	Path to the where CruiseControl is installed
Stop CruiseControl	CCConsole	Port number of where the JMX console is running
Show CruiseControl's UI / Documentation	CCServer	URL to CruiseControl's server
	CCPort	Port number used by the UI service

Table 1: Required information for drop-down list actions

CreateConfig and EditConfig action does not require information in this plugin's configuration file. However EditConfig action will open an editor on the file chosen by the setPath action.

Below is a class diagram shows the relationships among all the java classes of this plugin focusing on the executing order of the actions. Please notice that all classes in an Eclipse plugin depend on the activator class. This is not showed on the diagram. To keep the diagram clear, relations between action classes and Control are also not shown. Readers of this article please keep in mind that except the CreateConfig and EditConfig action, all the other actions

require paring from a XML file therefore they are related to the Control object.

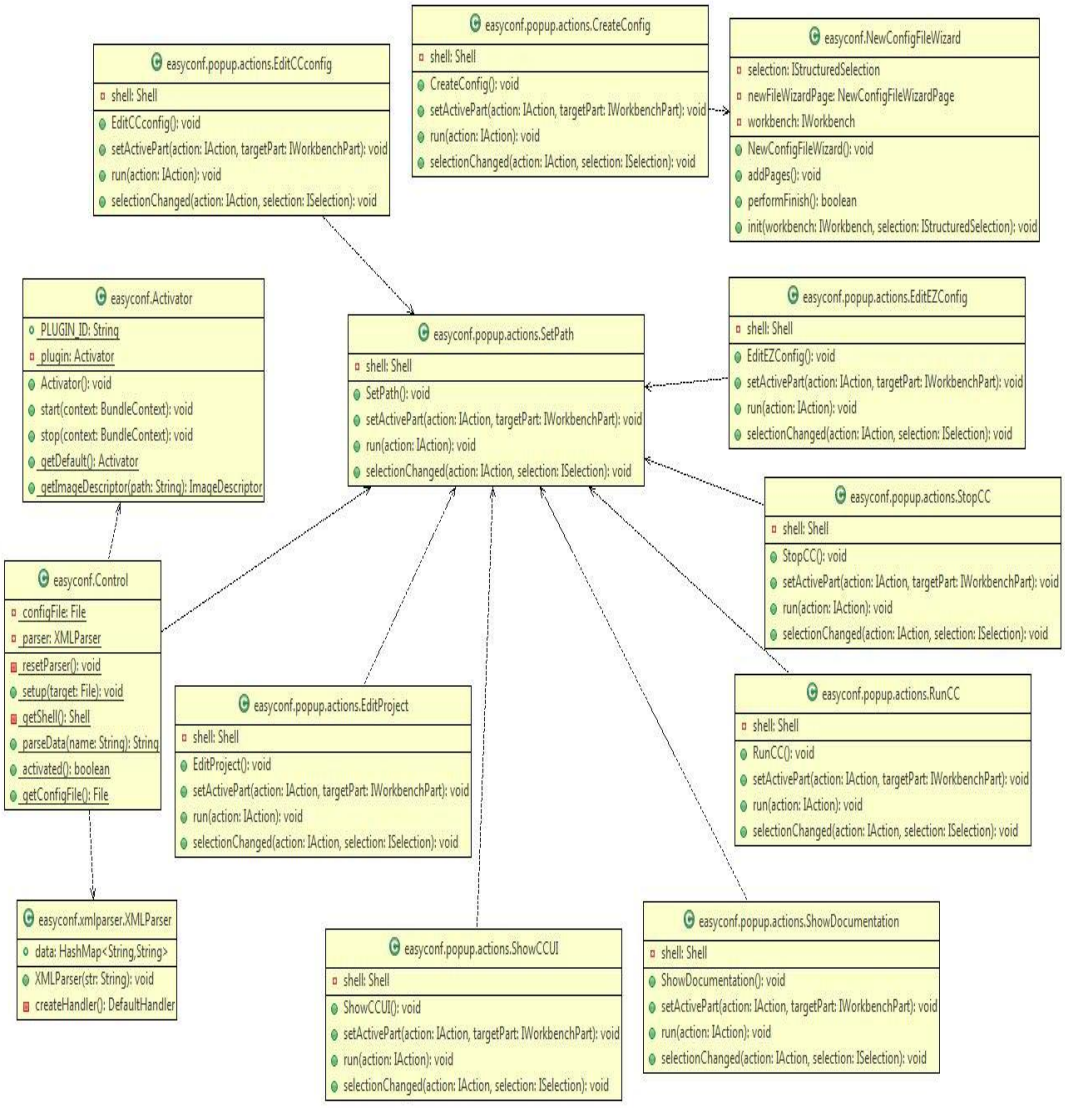

Figure 5: Class diagram of the Eclipse plugin in this project

Open CruiseControl's UI action and show documentation actions are in fact opening the links to the web services. If CruiseControl is not running they will still open the links but the pages will not be found. Therefore they depend on the RunCruiseControl action non-programmatically.

3.5 INTEGRATION OF THE DEBUGGING TOOLS

The CruiseControl distribution used in this project (Version 2.8.2) includes JUnit 4 library and PMD reports (PMD library is not included). It does not include support for Findbugs and will not recognize Findbugs reports.

To provide complete support of the two debugging tools: FindBugs, and PMD, their library files are extracted from the latest software distributions and are put into the lib directory of CruiseControl. Users can include the tests in their projects by defining and using the ant tasks.

Changes are made to CruiseControl's reporting and the user interface JSP for it to handle Findbugs reports. Similar to the way it handles JUnit 4 and PMD reports, it uses XSLT technology to convert the XML reports generated by Findbugs to Html-based report. The converted report is later merged into the general report which is presented to users by CruiseControl's JSP page.

4. VALIDATION

4.1 IMPROVEMENTS

By using the Eclipse plug-in, the efficiency of developing Java software projects are improved in following aspects:

Configuring CruiseControl is easier.

Instead of searching for configuration files in places on the file system each time the user wishes to add, modify, or remove a project managed under CruiseControl, the user now only needs to indicate paths to the files once and the information is saved to files. A configuration file represents a setting of the software. To apply a specific setting, the user now only needs to activate the corresponding file.

Configuration is in-place within Eclipse

Now a programmer can configure CruiseControl within Eclipse. The CruiseControl's User interface is accessible from a drop down menu provided by the Eclipse plug-in.

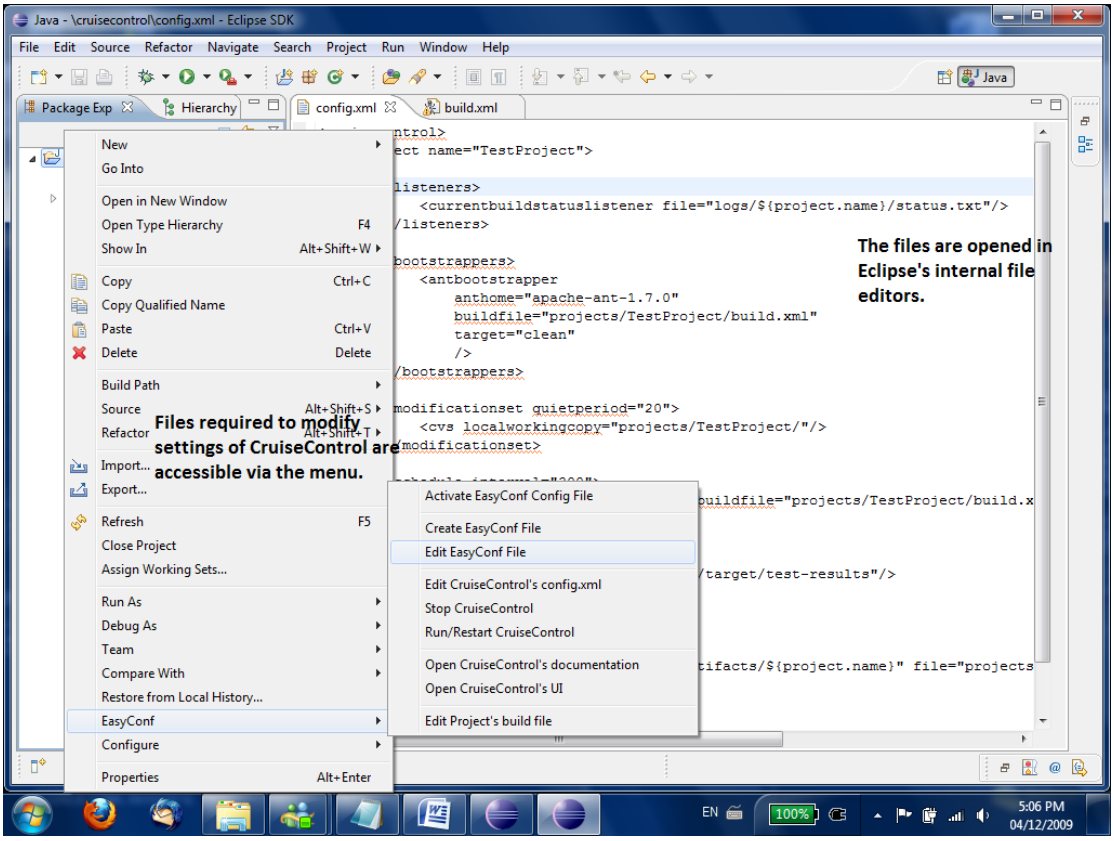

Figure 6: Validation: CruiseControl Configuration, part 1

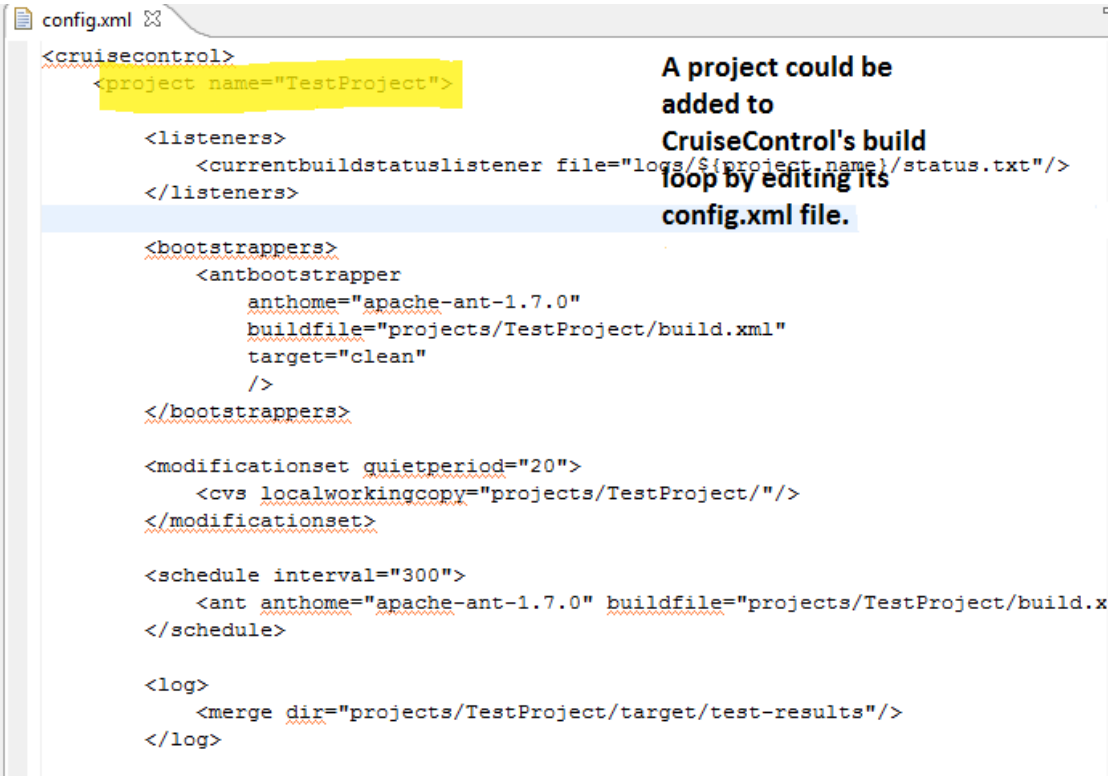

Figure 7: Validation: CruiseControl Configuration, part 2

As shown by above diagrams (Diagram 6&7), all works with respect to configuring CruiseControl could be done in the Eclipse environment using the buttons provided by this Eclipse plugin project.

The programmer is also able to start, restart, or shutdown CruiseControl within Eclipse and take the advantages of CruiseControl, which could help programmers in the following aspects:

Debugging software is more efficient

Traditionally the programmers have to manually run each test against a software project under development. Now CruiseControl will automatically build and test the project. Supporting CVS, CruiseControl will get the latest version from the CVS server and run builds and tests on defined time interval. The process does not require interactions between the software and a programmer. Thus, the programmers can now work on their project while CruiseControl does its works.

Programmers can also make several test plans, each in a different build file. Different test plans could be run against the software under development for different situations. Therefore testing is now more flexible.

Build and test reports are more informative and more useful.

CruiseControl collects results of the builds and the tests. The results are then be organized and published to the web server. These reports will provide useful information to the programmers for them to fix software bugs more quickly.

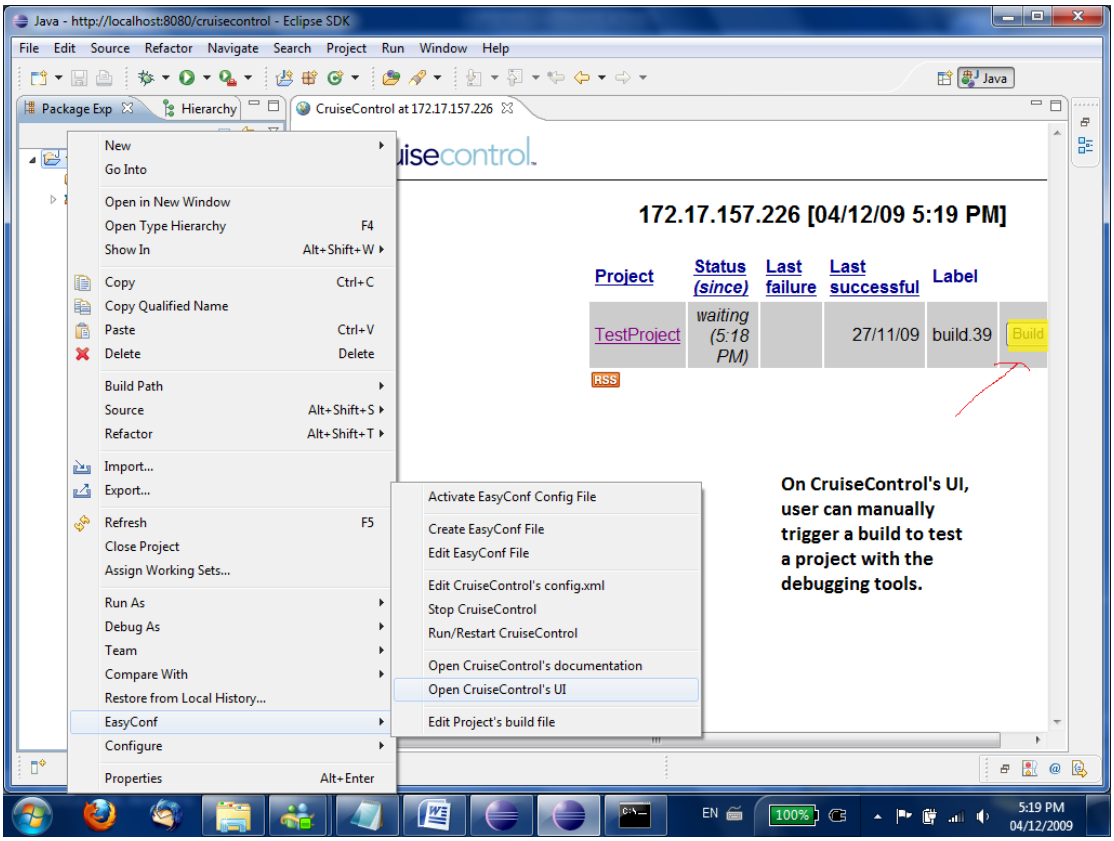

Figure 8: Validation: Debugging software projects, part 1

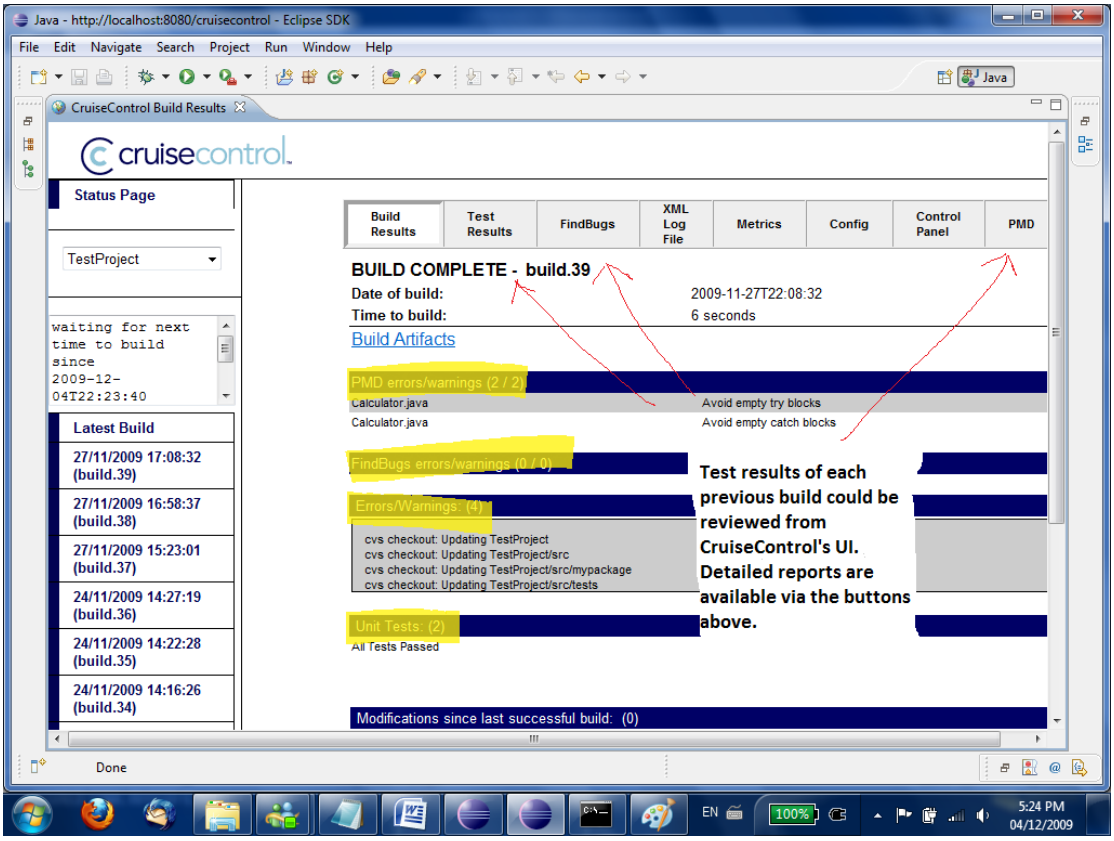

Figure 9: Validation: Debugging software projects, part 2

Diagrams above shows that user of this plugin could use it as a one-click debugging tool which provides in-detail reports of testing the software projects.

Below is a diagram shows the Eclipse plugin project successfully implemented functionality to fulfill the typical use of CruiseControl which is discussed in section 3.2:

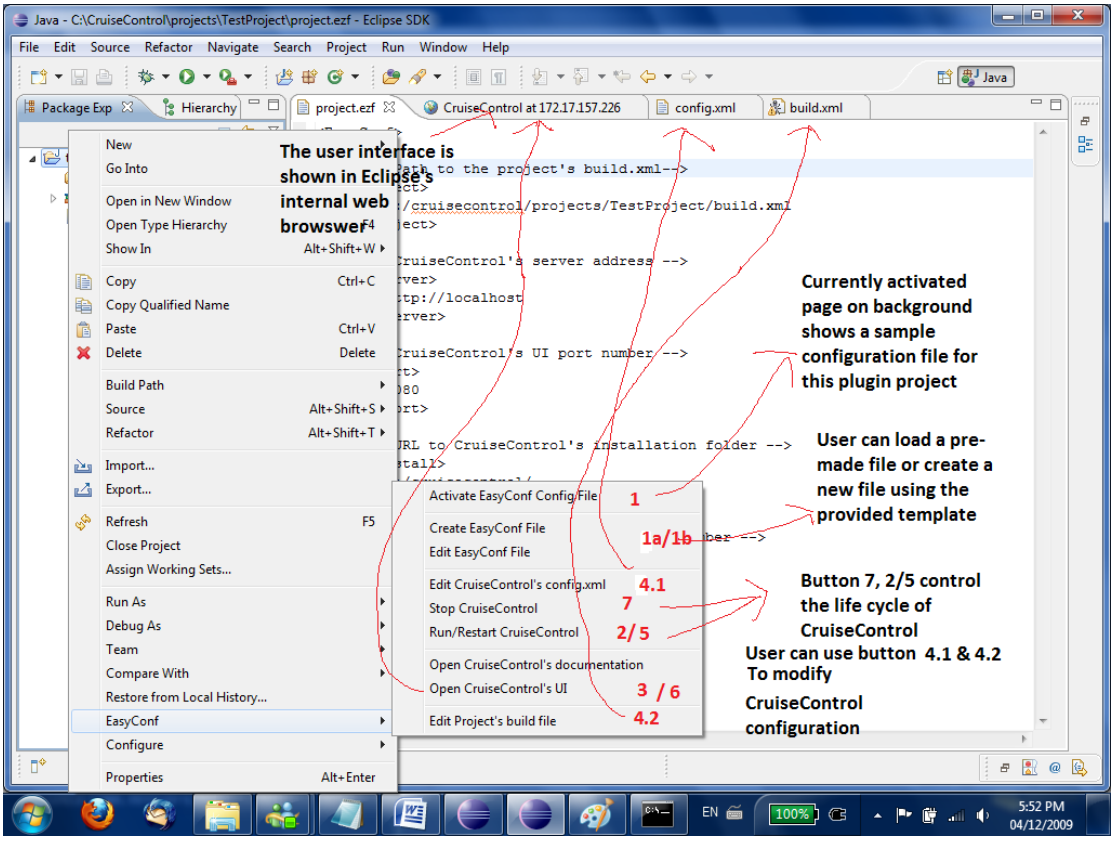

Figure 10: Validation: Typical use of CruiseControl inside Eclipse

The numbered buttons refer to the steps in the typical use. From the interface user must load an existing configuration file of this plugin project (1). If that file not exists, user can create such a file from the provided template (1a) and edit the contents of it (1b). User may then run CruiseControl (2). After CruiseControl is up and running, its UI could be opened (3). User could edit the configuration file of CruiseControl and build file of the project managed by CruiseControl anytime as long as locations to the files are provided correctly in the activated configuration file of this plugin (4.1/4.2). The changes could be applied then by restarting CruiseControl (5). User may want to rebuild the project immediately to see the results after the modification, which is

accessible from CruiseControl's UI (6). When the works are done user may leave CruiseControl running or shutdown the software (7).

4.2 COMPARISON WITH CRUISECONTROL CONFIGURATION TOOL

CruiseControl Configuration Tool is a Java webstart application for making management and configuration of CruiseControl easier and monitors software project managed by CruiseControl. The author of this project is Allen Wick. [15]

Comparing to CC-Config, this Eclipse plug-in project for easy configuring CruiseControl has several advantages:

First, it is an Eclipse plug-in. CC-Config is designed to be a stand-alone Java swing program. For developers who use Eclipse and CruiseControl, it is better to have the functionality of configuring CruiseControl in Eclipse. The Eclipse plug-in is also more portable than a program requires installation.

Second, it is easier to use. The user interface of the Eclipse plug-in is straight forward and does not require learning. A user could easily use it without looking into any guidelines.

Third, it is more than a configuration tool. The Eclipse plug-in is a part of this

project, which includes a pre-configured bundle of CruiseControl with supports of JUnit, FindBugs, and PMD as the debugging tools and the configuration user interface in Eclipse. It is ready to be used and, again, does not require extra learning.

On the other hand, CC-Config is a more powerful tool than this project. It provides graphical based configuration with configuration help in place so that someone who is not familiar with CruiseControl could use it. It also provides list of projects under management of CruiseControl with a name filter for searching names in a project. It enhances usability for CruiseControl users (Most likely a software company) who have a lot of projects under development. CC-Config is also a more general purpose tool than the Eclipse plug-in because it is a stand-alone program and it supports multi-platforms.

It is hard to draw conclusion that which project is better than the other as CC-Config is open-source and has been developed for a longer time and the Eclipse plug-in project is only at version 1. However it is obviously a trend that more and more tools are going to have their Eclipse plug-in version. From the home webpage we can see that the developers of CC-Config have a long term project for Eclipse plug-in. [16]

5. CONCLUSION

Not only for debugging the software but also for determining how well a software product meets users' requirement, testing is always a mandatory part in development of software projects. The concept of automatic integration is really helping programmers in terms of effectiveness of programming. CruiseControl as an open source automatic integration software is a great aid to Java programmers. Given that it is set up properly, CruiseControl will be a powerful tool for testing software products and projects under development.

This Eclipse plug-in project is dedicated to provide a solution to Java developers who use Eclipse and want to take advantage of CruiseControl. Although it may be too early to conclude that such a solution is adequate, it does provide utility which a programmer needs to use CruiseControl within Eclipse. By using the functionality of the Eclipse plug-in, time spent on miscellaneous works beside software development is saved, thus the productivity of the programmers is improved. For example, before using the plug-in there was a fair amount of repeated works in searching for specific configuration files. A programmer must leave the Eclipse environment to change the files and restart CruiseControl so that the changes could be applied. Now for the same task a programmer can simply activate a pre-configured file and all the files he wants to change are accessible in

Eclipse.

Programming this Eclipse plug-in project is a process of learning Eclipse and a practice of developing Eclipse plug-ins. It is also a study of human interaction with computer software in searching of the best way to improve work efficiency. It is a scheme of choosing the right user interface, the functionality to implement, and the way of a software product works so that users of the software could benefit from the achievement without working through the rigid process of learning. Within the time interval of developing this project and without make this project to the public for feedbacks, it is hard to draw conclusion that whether the scheme was well done. However, it is an experience gain and is helpful for developing software in the future.

Many open source software products are used in completion of this project. In development of this project they are studied and compared with many other software programs. A lot of knowledge is leant for a better understanding the world of open source software.

5.1 FUTURE WORK

The modified distribution of CruiseControl with the testing software is shipped to the users as a ready-to-use one-click debugging tool. If there is a new

release of any of the three debugging tools, the user may want to upgrade the existing ones in the bundle. It could be easy done by replacing the library files in the lib directory and they should work without a problem. However if users want to exploit the new features of later versions, they have to modify the reporting part of CruiseControl. This conflicts with the concept of designing this project because the work to put new features in the bundle is not easy for new CruiseControl users. For the future, an installation tool could be invented to take the parts needed out of the latest distributions of the software and the tools and integrate them into one. If required, a website could be set up and the bundle pre-built after each new version of CruiseControl could be downloaded from the website.

This project works well enough as a one-click debugging tool. However it is a weak tool for configuring CruiseControl. For version two of the software, one could study CC-Config mentioned above and try to improve this plug-in project with a fully functional graphic interface. It could provide sufficient help for a user who has little knowledge of Ant and CruiseControl syntax to use CruiseControl. It is also a great help if an editor for CruiseControl which provides keywords highlighting function could be provided to user.

CruiseControl could be very useful for developers who develop and test multiple software projects at the same time. Because this project is designed in

a way that each of the plug-in's configuration file is associated with a project, it does not allow user to run CruiseControl with more than one software project together since originally the plug-in is meant to help individual programmers or small team of programmers who are likely focusing on less than 3 projects at the same time. However to make it a better tool, the Eclipse plug-in project should allow adding multiple projects to CruiseControl.

It would be ideal if user is able to add a new debugging tool to CruiseControl as well as changing the reporting JSP for it to recognize and report the results of running the new tool for the future version.

The sole purpose of developing this Eclipse plug-in project is to help Java programmers and to improve their work efficiency. Developer of the future version of this project should see reviews of using this plug-in by the programmers and improve the usability of this software using the feedbacks.

REFERENCES

[1] CruiseControl.net, "CruiseControl Home", *CruiseControl Home*, http://cruisecontrol.sourceforge.net/index.html, Retrieved 16 Nov. 2009.

[2] CruiseControl.net, "Overview", *CruiseControl Home*, http://cruisecontrol.sourceforge.net/overview.html, Retrieved 16 Nov. 2009.

[3] CruiseControl.net, "The CruiseControl Build Loop", *CruiseControl Home*, http://cruisecontrol.sourceforge.net/main/index.html, Retrieved 16 Nov. 2009.

[4] Eclipse.org, "The Official Eclipse FAQs", *Eclipse Wiki*, http://wiki.eclipse.org/The_Official_Eclipse_FAQs#Java_Development_in_Eclipse , Retrieved 17 Nov.2009.

[5] Vogel. L, "OSGi with Eclipse Equinox - Tutorial", Vogella.de, http://www.vogella.de/articles/OSGi/article.html", Published 24 Nov, 2009. Retrieved 26 Nov, 2009

[6] Ant.Apache.org, "Frequently Asked Questions", *Apache Ant Wiki*, http://ant.apache.org/faq.html#ant-name, Published 16 Sep, 2009, Retrieved 17 Nov. 2009

[7] Chopra. V, Li. S, Genender .J, 2007. "Professional Apache Tomcat 6". Wiley Publishing, Inc.

[8] Codehaus Foundation, "Jetty @ Codehaus Wiki", *Jetty Wiki*, http://docs.codehaus.org/display/JETTY/Jetty+Wiki", Published 6 Oct 2009, retrieved 17 Nov. 2009.

[9] Wilkins, G. "Jetty vs. Tomcat: A Comparative Analysis", *Webtide*, http://www.webtide.com/choose/jetty.jsp, Published May 2008, Retrieved 17 Nov. 2009.

[10] Simpson, B. "JUnit 4.x How-to", *Axis Data Management Corp*, http://pub.admc.com/howtos/junit4x/, Retrieved 17 Nov. 2009.

[11] Harold, E, R. "An early look at JUnit 4" *IBM DeveloperWorks*, http://www.ibm.com/developerworks/java/library/j-junit4.html, Published 13 Sep. 2005. Retrieve 17 Nov. 2009.

[12] Rutar, N. Almazan, C, B. Foster, J, S., 2004, "A Comparison of Bug Finding Tools for Java", *Proceedings of the 15th International Symposium on Software Reliability Engineering, Pages: 245 – 256.*

[13] Vesperman, J. "Introduction to CVS", *O'Reilly Linux Dev Center.com*, http://linuxdevcenter.com/pub/a/linux/2002/01/03/cvs_intro.html, Published 3 Jan 2002. Retrieve 17 Nov. 2009.

[14] Lee. K. A., 2006, "IBM Rational, ClearCase, Ant, and CruiseControl: The Java Developer's Guide to Accelerating and Automating the Build Process", IBM Press.

[15] CC-Config, "CruiseControl Configuration Tool Home", CC-Config.SourceForge.net, http://cc-config.sourceforge.net, Retrieved 18 Nov. 2009.

[16] CC-Config, "CruiseControl Configuration Client Changes", CC-Config.SourceForge.net, http://cc-config.sourceforge.net/todo.html, Retrieved 18 Nov. 2009.