The Eclipse Grading System Plug-In

A Major Qualifying Project Report:

Submitted to the faculty of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

by

Scott Kolodzy

Kenneth Louf

April 24, 2008

Professor Gary F. Pollice, Advisor

1. Assignment Submission
2. Automatic Grading
3. Eclipse Plug-in
Abstract

Assignment submission and grading is not always the easiest of tasks. For some, it can be a frustrating, error-prone process. Students can incorrectly export submitted assignments, resulting in grader frustration and in some scenarios a zero given for what would otherwise have been a passing assignment. In this project, we developed a plug-in for the Eclipse Integrated Development Environment to perform assignment submissions to an assignment submission server that does automated grading, easing the strain placed on the graders of programming assignments.
Acknowledgements

We would like to thank Professor Gary Pollice for advising this project. Without his guidance and direction, this project would not have been as successful as it was. We would also like to thank the other MQP groups we worked with for helping us with problems we could not overcome on our own, and for providing us with feedback and testing our plug-in. The Web-CAT team at Virginia Polytechnic Institute also deserves thanks for developing a submission server that met our requirements and provided ideas for future plug-ins.
Contents

INTRODUCTION ........................................................................................................................................ 2
  1.1 Problem........................................................................................................................................... 2
  1.2 Requirements................................................................................................................................... 3
  1.3 Solution Approach .......................................................................................................................... 4

BACKGROUND........................................................................................................................................... 4
  2.1 Eclipse IDE ..................................................................................................................................... 4
  2.2 Submission Servers ......................................................................................................................... 5
  2.3 Web-CAT Support for Testing Submissions .................................................................................... 7
  2.4 Existing Plug-Ins for Assignment Submission .................................................................................. 8

METHODOLOGY ....................................................................................................................................... 8
  3.1 Existing Web-CAT Submission Plug-In........................................................................................... 8
  3.2 Features .......................................................................................................................................... 9
  3.3 Approach....................................................................................................................................... 11
  3.4 Testing........................................................................................................................................... 12

RESULTS.................................................................................................................................................... 13
  4.1 Issues Encountered ....................................................................................................................... 13
  4.2 Assignment Submission Process ................................................................................................... 14

FUTURE WORK ....................................................................................................................................... 20

SUMMARY ................................................................................................................................................. 21

REFERENCES ........................................................................................................................................... 22

APPENDIX A: HELP FILES.................................................................................................................... 23
  A.1 Getting Started.............................................................................................................................. 23
  A.2 Connecting .................................................................................................................................... 25
  A.3 Submitting an Assignment ............................................................................................................ 27
  A.4 Setting Preferences ....................................................................................................................... 31
Introduction

The Eclipse Grading System Plug-in (EGSP) is a tool that allows students to efficiently submit assignments and receive automatically graded results for homework assignments from within the Eclipse Integrated Development Environment (IDE). The Computer Science Department at Worcester Polytechnic Institute currently has no system in place for submission and automated grading of homework assignments from within Eclipse, making the submission process difficult. This plug-in would replace existing methods which include e-mail, turnin[5], and printouts. These methods, while reliable, do not allow for a student or grader to submit or grade assignments very efficiently or easily. With the Eclipse Grading System Plug-in in place, a server on campus running Virginia Polytechnic Institute’s Web-CAT[4] software would accept assignments submitted directly from the Eclipse IDE. Students could submit their work through an intuitive interface, and graders could more easily retrieve and grade those assignments. This submission and grading system will make the jobs of both students and graders much easier.

1.1 Problem

There is currently no standard method to submit homework assignments done in Eclipse. Some courses use e-mail, and others use turnin. The issue with these methods of turning in assignments is that there is a large chance to introduce user error. Users can introduce problems with exporting or archiving of their assignments, preventing the assignments from easily being imported into another workspace. Graders can not recover the data from the assignment without significant effort, and as a result the grading
process is so slow that results are not able to be given in a timely fashion. In other cases, an e-mailed submission can fail to be sent, especially if the address required to submit it is not clearly specified somewhere. There can also be issues with e-mail inbox sizes and attachment size limitations. Without a clearly defined set of steps that every user follows, submission and grading of assignments becomes a great burden on graders, which could be avoided. The problem lies mostly in the nonstandard submission processes.

1.2 Requirements

There were a few requirements that had to be met for this project. The assignments submitted would all be done in Eclipse, so the deliverable must be a plug-in. This plug-in should have an intuitive interface, looking similar to interfaces already present in Eclipse’s file exportation feature. It should also have its own view within the Eclipse workbench, and the ability to clearly identify what the user is doing as well as where in the process the user is, when submitting an assignment. The requirement of having graded results returned was optional, but implemented due to the greatly improved feedback it would provide.

On the more technical side, the plug-in must be able to handle the Web-CAT server at the very least, and support background file transfers. It must also automatically package the assignment files in a standard format, allowing for user-friendly submission of assignments. The files submitted also must be able to be correctly imported into Eclipse for a grader to grade manually, which is the purpose of the plug-in in the first place. There was also a requirement that it be extensible, for future support of an almost endless array of protocols, packagers and servers.
1.3 Solution Approach

Approaching the problem, we needed to identify possible solutions. Research was done into existing server and client technologies, and a best choice was found. Once we found a server to accept submissions, we then looked at the submission system for that specific server. Using existing technologies as a base to work off of, we wrote our own plug-in to interface with the packagers, protocols, and file types supported by the server we chose.

Communications to and from the submission server were in a standard format, which was taken from the previous plug-in that Virginia Tech had made to work with their server. We reused this networking information and concentrated on improving on the shortcomings of the plug-in that was already developed.

Background

In this section, we will discuss the background in researching and developing the plug-in, and coming up with an implementation that would meet the requirements and solve the problem. This information includes the Eclipse platform, different submission servers and the ideas behind them, and the preexisting plug-ins.

2.1 Eclipse IDE

The Eclipse IDE[2] is a widely used tool that the students at WPI use for coding in Java and C++. The issues with this tool related to the problem at hand include the ability to export files in many different ways, creating plenty of chances for the user to introduce errors and create files that cannot be easily properly imported again. The
Eclipse platform is very powerful in other ways as well. It has the ability to use Subversion and CVS repositories for code control, as well as tools to easily resolve version conflicts. A user can also edit multiple files at a time, and the layout of the package viewer makes object-oriented programming across multiple packages easy and fairly painless. Most importantly, it is highly extensible, which allows plug-ins to be written for the IDE and loaded on runtime. This is the primary development tool used to develop plug-ins for itself. For this reason, we used Eclipse to develop our final plug-in deliverable.

2.2 Submission Servers

There are currently a few submission servers available, including Web-CAT[4], Epaile[1], and Kassandra[3]. Each of these has their benefits and drawbacks. Kassandra, while having papers published on it, was not found to be easily available for general use, nor available for Java assignments. While the papers were readily available, and the features and specifications matched what WPI was looking for, there was no way to actually download the code and host a submission server. Kassandra supported the ability to submit assignments and have them graded, with correct solutions automatically assigned to the user. The assignments are chosen to be graded by calling the Kassandra program, specifying an assignment, and then uploading, testing, and returning grades. It tries to give feedback to the user, and will attempt to specify where exactly the student is having issues. Kassandra supports anonymous submission via a user ID given to a student by the instructor, as well as submission with full user accounts. Inquiries in progress may be made using either user tracking method, so a student can check previous assignments and grades.
Epaile, a program developed by a graduate student at Basque Country University in Spain for his final degree project, was another option briefly looked into. The main problem with Epaile was that while it was an assignment submission server with automated grading, it was still in rough beta and only maintained by an individual. It was more of a personal project, rather than an officially sponsored project from an academic or business institution. It does include the ability to grade and accept Java submissions and other languages, and includes assignment opening and due dates. There are limitations on the size of submissions accepted, so a malicious user may not overload the server. There is also a limit on runtime for a program, so an infinitely looping program will not break the server. Late assignment submissions can also be optionally disabled. Student assignment submissions and results are archived independently, and graded immediately or at a set date.

Web-CAT, an open-source project licensed under the GNU Public License (GPL) that is being continuously developed by the Department of Computer Science at Virginia Polytechnic Institute, was our choice for this project; it has many features of the others and more. It has an easy to use web interface, with submission available via the web interface as well. The server takes submissions via http file transfers also, allowing for submission from other external tools. The server advertises its’ security, which includes plug-in based authentication, which can be extended to use a university’s own authentication system. There is also detection of erroneous programs preventing infinite loops, and detection of malicious code to prevent exploitations of the server software. Data is stored in a relational database, for extra security. The server was written in Java and created using Apple’s WebObjects, a web application framework and server, making
it very portable and will run under any compliant server container. In this specific case, the server container is Apache Tomcat. Web-CAT is also flexible, allowing for plug-ins to be added for automatic grading of code in C++ and other languages. It has been used in the past to handle submission of Java, Scheme, C++, Prolog, Standard ML, and Pascal. It also has a system for manually grading student’s assignment submissions, in addition to automatic grading. We chose this server, as it had the most amount of useful features, was free to download and use, was easy to install, and performed the functions we were exactly looking for.

2.3 Web-CAT Support for Testing Submissions

The Web-CAT server that we decided on supports automated grading, as stated above. Automatic grading via Web-CAT has three distinct methods: a code correctness score, a test completeness score, and a test validity score. The code correctness score returns a score based on the number of a student’s own tests the submitted code passes. The test completeness score is a score based on how much code is covered by the test cases. The test validity score is given based on how many of the tests a student’s code passes. These validity tests can be provided to the server by an instructor to test the submitted code against.

When an instructor creates an assignment on the server, he (or she) can specify a required files option that will force submitted assignments to have specific files present in order to be considered valid. These files are required, with specific function names and class names, so that the automated JUnit tests can be run. These tests can also be run with linked-in classes, as provided by the instructor. There is a very flexible control over what exactly is run, and how.
2.4 Existing Plug-Ins for Assignment Submission

The Web-CAT project includes an Eclipse plug-in already built for submission. After using the plug-in to obtain a feel for the usage flow, it was easy to see that there was a lot of room for improvement. Primarily, the usage of this plug-in was not very intuitive, and it was very limited in what exactly it could do. There was a big problem with the interface, in that it could not easily guide a user through the process without having read documentation beforehand. Often, students will not read instructions before trying to use a plug-in or any other piece of software, so this was something concerning that we knew should be improved upon in our deliverable. The documentation was also poor, and the help was hard to find and ultimately not very helpful. These were also things that we looked to improve in our own plug-in.

Methodology

The plug-in provided with the Web-CAT server was not up to doing the job we had envisioned it doing, so we needed to use what we could and improve on everything else. Using the knowledge we had gained of the servers and their capabilities, we had a good idea of what we could do with the plug-in and how it would improve over the one provided currently with the Web-CAT software. This section will discuss our methodology and how we made our plug-in into what it is now.

3.1 Existing Web-CAT Submission Plug-In

The existing plug-in for Web-CAT had a lot of valuable information, despite the shortcomings we identified in our research. Primarily, it gave us ideas of how to
construct our plug-in, allowing for easy modification of code. The existing plug-in was very heavily object-oriented, with many interfaces and classes, allowing for a flexible system to be developed. This architecture was adapted for use with a more robust version of the plug-in. First and foremost, we needed to find out how to interface with the server. This involved trying to find an API or documentation on how to connect to the Web-CAT server to submit data.

Upon looking, we found that there was no API for interfacing with the server specified. The server’s protocol was implemented in the plug-in, but it was not documented very well. As a result, we reused some of the code for networking and transformed it to allow our new plug-in to able to communicate with the Web-CAT server. This gave us a code base to work off of, and having the ability to send assignments to the server from the start helped greatly in debugging and knowing we had networking working correctly. This code base also provided us with the ability to make the plug-in extensible.

3.2 Features

As a part of our plug-in, we wanted to enhance the features already present in the existing plug-in, as well as add in a few features that the existing plug-in did not have.

The first feature we added was the ability to open a view in the Eclipse workbench, allowing for easy access to the list of available assignments once downloaded from a server. This makes it clear to a student which assignments are available, and makes submitting to a specific assignment possible through alternative methods than originally available. This makes it much easier to tell what assignment is
being submitted, and with having the EGSP possibly spanning multiple courses, keeping homework for each course separate as well.

One of the features we enhanced was a wizard to submit assignments with. This wizard would have the ability to selectively submit files for assignments, as well as submit to an assignment after receiving a definition file from the server. It would improve upon the existing wizard by being more informative to the user and by being more responsive to submission errors. This should mirror the Eclipse interface in some ways for ease of use. An example of this is shown below.

Figure 3.1
This form continues throughout the interface, which will be shown more in detail later on, with an assignment submission walkthrough, and in the appendix with the attached help files. The wizard is reachable through a command in a contextual menu developed within the aforementioned view, or from a command in the context menu of the Package Explorer built into Eclipse.

The plug-in we developed also has the ability to be extended as needed to handle other server types, transfer types, and archive types. With extension points, this becomes a much more powerful tool than previous plug-ins because of the ability to leverage other servers, protocols, or archive types. This allows for many future expansions of the Eclipse Grading System Plug-In, with future WPI projects possibly building on its code.

3.3 Approach

To approach the problem, we wanted to reuse as many of the back-end things that were working well enough, and put them into the current plug-in. The main portion that was reused was the networking code. There is a specific way to package and submit an assignment to Web-CAT via background http transfers. The problem was this method was poorly documented, and trying to follow and understand all of the code included in the previous plug-in was not easy to do. Instead, we decided we needed to use the important parts of the networking code, and ported it to the EGSP. This allowed us to focus on making the user experience more enjoyable. Since we wanted a more user-friendly experience, we decided making both a view and a right-click context menu would be the best option to provide a complete, intuitive experience. After having used Eclipse for years, we both knew that there were differing opinions on whether right-click
context menus or views were more useful. Since there is no decisive answer, we implemented both.

Once we had context and view menus working, we then made sure that individual assignment submission was possible. This involved a testing process that we will visit in the next section. After working out bugs, we started work on the wizard for file submission. The wizard provides a series of steps for easy, user-friendly assignment submission with minimal effort. A submission process usually follows these steps:

1) Right-click on the assignment to submit to in the Assignment View
2) Select the assignment and provide server authentication (log-in) information to the plug-in
3) Select the files you want to submit to the server
4) Verify your files being submitted, and verify ability to submit
5) Submit the assignment

These steps are simple and intuitive when using the interface, as each step in the process tries to mirror Eclipse interfaces already in place. The wizard when completed was tested extensively to make sure it was exporting assignments in a gradable format, and when deemed complete was graduated to a beta test.

3.4 Testing

Testing for this plug-in was all done manually, as there was no way to automate the tests. The tests followed a standard format, and followed the same steps in a typical scenario each time. Primarily the testing was done by our team, and most of the show-stopping bugs were found and ironed out because of this. We did not move on to a next stage without making sure all critical bugs were removed.
In addition to heavy development testing, there was a planned beta test of the plug-in as well. The introductory Java course taught by Professor Neil Heffernan planned on using the EGSP for assignment submission. After discussions with a Teaching Assistant, we attempted to have a full beta set up. Unfortunately due to time constraints, the class was not able to set up a Web-CAT server for use with the plug-in we developed.

A fallback method was having persons other than ourselves test the plug-in by using it to submit an assignment to a submission server we had set up for testing on a local, decentralized machine. These students were selected from persons we knew and were familiar with, but included students outside of the Computer Science department. These tests went very well, and showed a great improvement of usability over the previous plug-in that came with Web-CAT. It was then that we decided that our interface was good, and that all that was left was to iron out bugs that were present with the submission and file wrapping component of the plug-in.

Results

In this section we will discuss issues encountered, and review an assignment submission process. Following the results will be a section on future work.

4.1 Issues Encountered

Overall, our plug-in was a success. The very first issue encountered was that there was no documentation on how to submit an assignment to the Web-CAT server over an http connection outside of the web interface. We knew Web-CAT supported this
method of submission, but the only reference we had was the plug-in already developed by the Web-CAT team. This plug-in had little to no useful documentation, and there were also few comments in the code itself. Dissecting the plug-in to use the networking and submission code was a very large part of our earlier work. This slowed down the initial process, causing us to fall behind schedule at first. Once the networking code was implemented, things moved much quicker.

The main issue we ran into was the lack of a full beta test. We had one organized, and were discussing the possibilities with Professor Neil Heffernan’s Teaching Assistantss for giving us feedback, but unfortunately due to time constraints it did not happen. We also tried to consult with Professor Heineman about his Software Engineering course, but at that point it was too late to get a beta test fully set up to obtain meaningful results.

4.2 Assignment Submission Process

In this section, we will review an assignment submission process, using graphics from the help we have provided with the plug-in.

First, we want to connect to the Web-CAT server to see what assignments are listed, and we do that in the Assignment View which is opened in Eclipse’s main window:

![Figure 4.1: Connection Dialog](image)
We then provide log-in information, which can be used to authenticate a user even before assignment information is downloaded:

![Server Connection Information](image)

**Figure 4.2: Server log-in dialog**

Next, we receive a list of assignments, and can submit them within the Assignment View.

![Assignment Submission View](image)

**Figure 4.3: Assignment Submission View**

Once we specify an assignment to submit, we then enter the assignment submission wizard, which will check which assignment you want to submit.
Once you have selected an assignment to submit, there is a dialog to select the files you wish to submit.
Once these files are selected, and the OK button is pressed, if you are missing required files the system will alert you.
Once the required files are included, you will then have a confirmation dialog to verify you are submitting the files you intended to submit.

![Assignment Submission Wizard](image)

**Figure 4.7: Submission Confirmation**

Once you have verified the submission, you submit it to the server. A confirmation dialog will let you know if your submission succeeded or failed.
Afterwards, if the assignment was set up with automated grading, you will receive your results directly in Eclipse, with information available on the page telling where there are errors in your assignment.
These results can be given a set number of times, based on settings the instructor provides to the server while setting up an assignment definition. This allows for some revisions to be made, but not for abuse of the system where a student keeps submitting an assignment until a grade of 100% is achieved.

**Future Work**

Future work can certainly be done on this project. There could be future MQP projects to develop plug-ins for the Web-CAT server, allowing for WPI’s STUDENT Domain or CS Department accounts to act as an authentication mechanism for students for the Web-CAT server. This would make setting up a course much easier, as you would not have to submit a list of students and distribute log-in information to all of
them. The Computer Science Department log-ins would also be a good choice, since they are by-request accounts and students not in a course with a need for access to the server would not have access. This authentication method would also remove a glaring issue from the Web-CAT server, in that while stored in a secured relational database, the passwords for users are stored as plain text values.

There also could be a server developed to WPI’s specifications for an MQP. Web-CAT may not be exactly what WPI is looking for, and developing a submission server would make a suitable MQP project. A closely related project to this one would be extending the EGSP to use this new server for assignment submission. Since the plug-in is extendable, this would not require an entire re-write of the code already developed. There could also be added support for different packaging methods, or other submission protocols. These all could tie in with a server that WPI develops.

Summary

The project was a success, delivering a plug-in for Eclipse that will submit assignments to a server on campus, which will automatically grade them and return results. This server can be hosted by the Computer Science department full time, and will make submission of assignments much easier. A full beta version of the plug-in will be available at the end of this project, and hopefully the system will be adopted. There is room for improvement in the form of future projects, and time will show the true value of having an automated submission plug-in.
References


Appendix A: Help Files

A.1 Getting Started

Getting Started with the Assignment Submitter

This section will cover opening the Assignment View to use in Eclipse.

Opening the Assignment View

With the Webfoot plug-in installed and viewing in Eclipse:

1. Go to Window/Show View/Other...
2. Select Assignment View under the Webfoot Folder

3. You will now see the Assignment View open on your screen

You are now ready to proceed to the next step!
A.2 Connecting

Connecting To a Server

This section will cover connecting to a server using the Assignment Submitter. There are two methods for connecting to a server:

1. through the Assignment View, and
2. through the context menu of the Package Explorer or Navigator

Connecting through the Assignment View

1. Right click in the Assignment View window
2. Select 'Connect to Server...'

![Connection dialog](image)

3. Fill in the server URL and the server type (probably provided to you by your instructor), your login name, and password in the dialog as shown below. Some of the information may be filled in with default values taken from your preferences, which can be set for convenience.

![Server connection information](image)

4. Click OK to connect

Connecting through the context menu of the Package Explorer or Navigator

1. Select some files/folders/a project to submit from the Package Explorer or Navigator. NOTE: multiple files/folders must be from the same project in order to be submitted.
2. Right-click on any of the selected items and select 'Submit Assignment...'.

3. Fill in the server URL and the server type (probably provided to you by your instructor), your login name, and password in the dialog as shown below. Some of the information may be filled in with default values taken from your preferences, which can be set for convenience.
A.3 Submitting an Assignment

Submitting an Assignment

This section will cover submitting an assignment using the Assignment Submitter. There are two methods for submitting an assignment:

1. **through the Assignment View**, and
2. **through the context menu of the Package Explorer or Navigator**

Submitting an assignment through the Assignment View

1. Select an Assignment from those displayed in the Assignment View window
2. Right-click on the selected assignment and Select 'Submit Assignment...'

3. **Complete the Assignment Submission wizard**
4. You are done!

Connecting through the context menu of the Package Explorer or Navigator
1. Select some files/folders/a project to submit from the Package Explorer or Navigator. NOTE: multiple files/folders must be from the same project in order to be submitted.
2. Right-click on any of the selected items and select 'Submit Assignment...'

3. Complete the Assignment Submission wizard
4. You are done!

Completing the Assignment Submission wizard
1. If you have not already selected an assignment to submit from the Assignment View, select an assignment here. Also, make sure that the username and password fields are filled in and click 'Next >' to continue.

2. If you haven't already selected some resources from the Package Explorer or Navigator, select some files/folders/a project to submit from the next dialog as shown below and click 'OK' to continue. NOTE: multiple files-folders must be from the same project in order to be submitted.
3. If your submission is missing required files, you will encounter a dialog box as shown below. Answering 'Yes' will add the required resources to your submission and continue with the wizard, while answering 'No' will not add the resources to your submission and return you to the assignment selection screen of the wizard.

4. A summary screen will display any problems with your submission; you must resolve these before you may submit your assignment. Note that you may click '< Back' to go back and make any changes you would like in the wizard, or choose
'Cancel' to exit the wizard at any point. Click 'Next >' once these have been resolved to make your submission.
5. Your submission is made and any errors are displayed in the results screen. Click 'Finish' to finish the wizard, return to the Eclipse workbench and view the results of your submission, or'

Congratulations, you have submitted an assignment using the Assignment Submitter!

A.4 Setting Preferences

Setting Assignment Submitter Preferences

This section will cover setting preferences for the Assignment Submitter. These preferences provide a convenient way to set default values for use in the plug-in, so that you don't have to manually enter similar values every time you use the plug-in.

1. From the 'Window' menu in Eclipse, select 'Preferences...'
2. Under the 'Webfoot' tree item, select 'Grading System Plug-in'
3. You have the option to fill in a server URL and a server type (probably provided to you by your instructor), and a login name, as shown below.

4. Click 'Apply' to save your input and set other preferences or click 'OK' to save your preferences and exit the preferences dialog.