CodeHandIn: An Automated Programming Assessment Package for FLO

Author: Samuel Deane, dean0109@flinders.edu.au Supervisor: Associate Professor Paul Calder

Aims and Objectives

The aim of this project was to continue development of a plugin/module for the Moodle Learning Management System (LMS) which will be integrated into Flinders' own Moodle System, Flinders Learning Online (FLO), for use in future programming topics.

Background

Automated Assessment of student programs has the following benefits:

- Frees tutors and lecturers from marking and answering repetitive questions
- Aids students in their assignments, giving them direction, strengthening their skills and introduces the test first methodology

However many Automated Assessment system that have bene produced have flaws such as:

- Being designed for a single purpose and not integrating with Learning Management Systems like FLO
- Based on Limited internal development, often as part of an academic exercise
- Not versatile across different programming areas leading to reinvention for different topics

To solve these problems the Codehandin module for the Moodle (generic FLO) Learning Management System (LMS) was developed last year (2013). The module provided support for the management of a programming assignment type and allowed automated testing of student programs.

This project continues the development of this system, improving the system by adding new functionality, extra interfaces and revising the overall design.

System Improvements over the 2013 version

- Updated System architecture (right), that splits the codehandin module into a assignment submission and a assignment feedback plugin removing the need to replicate the assignment features
- Background architectural improvements (not shown here)
  - Reworked database … uses assignment variables rather than copying
  - reworked web service … full JSON Restful web services
  - Centralised code … all contained in the web services shared library
- New IDE integration – a NetBeans Client (figures 2 and 3)
- Better Sandboxing – added support for the external CompileBox compiler
- Expand Moodle Web interface – allows management (update and deletion) rather than just creation of Codehandin Assignments (see figure 1 bellow)
- Better programming language support – languages added through DB not code
- Increased Documentation – will aid in future moodle plugin development

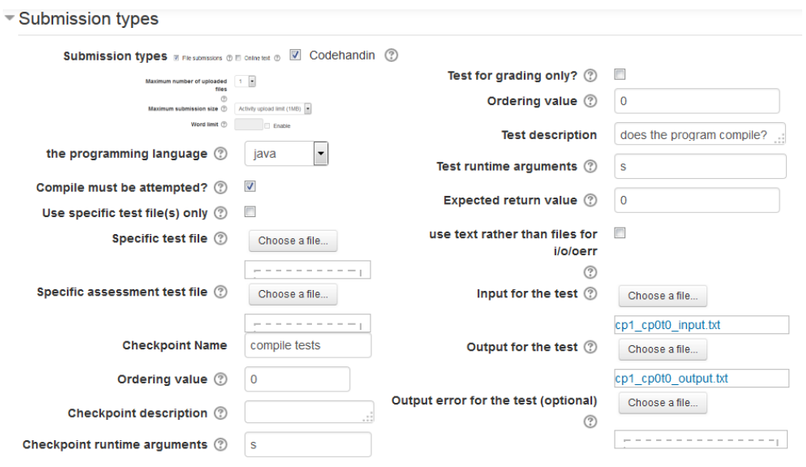


Figure 1: Submission plugin – CodeHandIn assignment setup
Note: seen as long and wide single column on the assignment setup page

1. CodeHandIn Submission Plugin

Creates Codehandin Assignments: Defines the tests and checkpoints (groups of tests) and other information to make a codehandin through the web interface (see figure 1 middle).

2. Client plugins

Manage and submit CodeHandIn Assignments for testing remotely:
A native Java client plugin (used in a NetBeans plugin see figures 2 (bottom left) and 3 (bottom middle) connects to the web service to allow staff to manage codehandin assignments and students to download and submit code for testing.

3. CodeHandIn web service

Tests and manages CodeHandIn submissions and assignments: Does the actual testing and provides functionality to manage codehandin assignments through the definition of external services and a global library for general use.

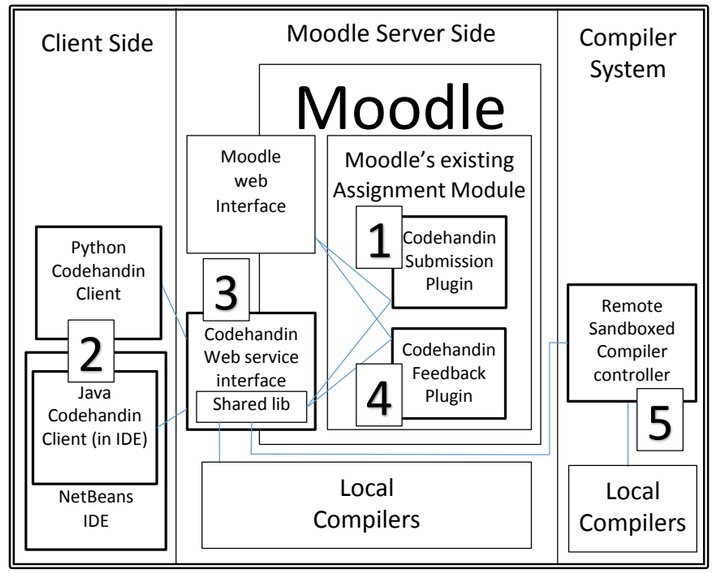
4. CodeHandIn Feedback Plugin

Feedback for submissions: Assignment grades integrate with the gradebook and allow an additional style mark as well ass the usual text feedback.

5. Remote Sandboxed Compiler controller

Compile submitted programs safely: Submitted programs are compiled on a remote sandboxed Compiler controller (compileBox used) rather than on a local compiler.

Updated System Architecture



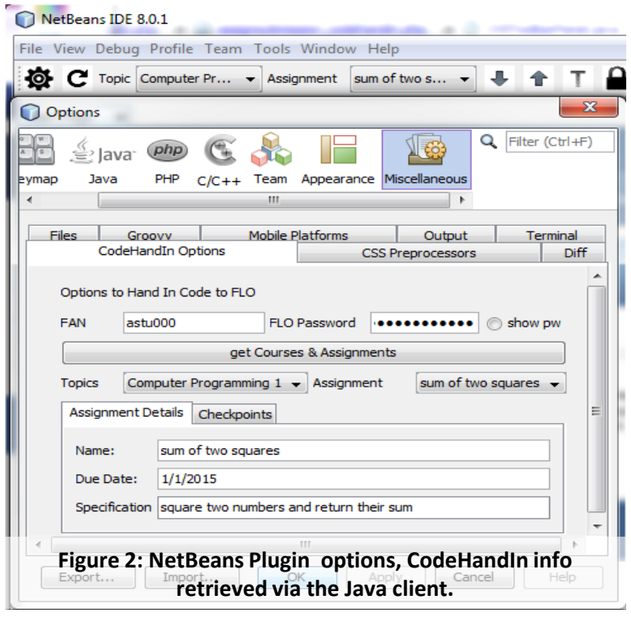


Figure 2: NetBeans Plugin options, CodeHandIn info retrieved via the Java client.

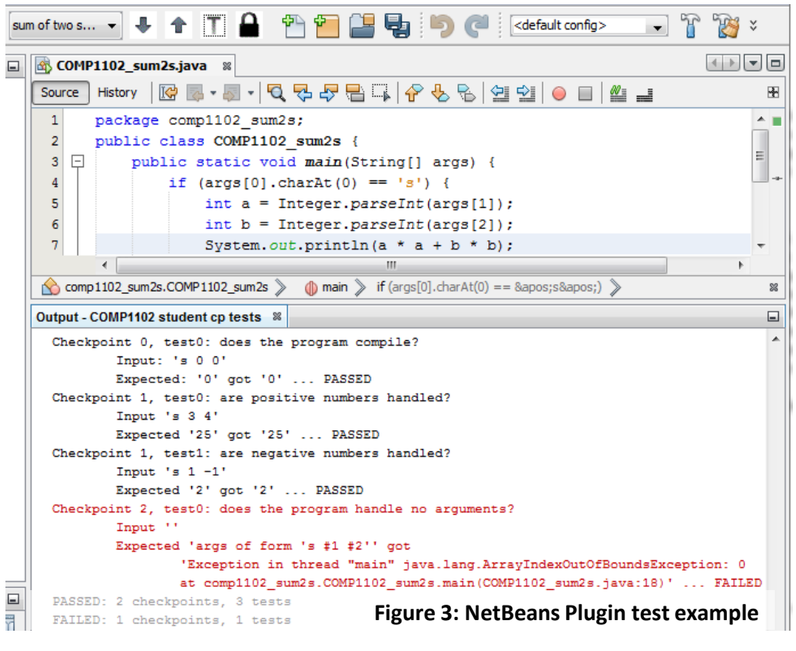


Figure 3: NetBeans Plugin test example

Conclusion

This project represents a significant improvement in the Moodle CodeHandIn package; predominately revising the architecture to that of a plugin rather than a module and adding more management options (including a more detailed creation GUI, a external sandbox compiler and a NetBeans plugin that can handle downloading, uploading, submitting and testing of codehandin assignments.

Future work

A lot still remains to be done, including:

- Split CodeHandIn setup over multiple pages (split figure 1)
- More testing and start using inbuilt Moodle testing tools
- Better use of the NetBeans API in the NetBeans plugin
- Introduce batch testing in the feedback plugin
- Make assignment reusable and copyable (stubs added)
- Make compiler controller run more than 1 test per compile.
- Open the source code of the package for external development

References and further information

Information on the Moodle Learning Management system can be found at https://moodle.org/
Details for the original Honours project authored by Jonathan MacKenzie are available at https://wiki.csem.flinders.edu.au/bin/view/CSEMThesisProjects/ProjectMack0242