

Report Two - Change

Michael Cook, Azalea Raad, Caroline Anjorin, Soroush Karimi, Andrew Lim

November 13, 2008

Overview

The project aims to develop a tool for automatically generating comprehensive test programs for a user-specified Java compiler. The user may specify the nature of the tests, which language constructs are investigated, and how complete the tests should be. The tests are run on the specified compiler, as well as an oracle compiler which is used as verification. The program reports on whether the user's compiler matched the output of the oracle.

Progress

The first iteration was due to conclude on Monday 3rd November, and had the following requirements:

- The GUI will be partially implemented, to allow a user to start a new project, add classes to their project and then execute the test suite they've constructed. Some configuration options will be available. The results will be displayed in a partial summary, with most options and formatting disabled. This will likely be in the form of a simple dump of results out onto the screen.
- The code generation will implement its first approach to test generation, focusing initially on programs including classes only - that is to say, no more detailed components such as fields or methods may be generated. The initial approach will use a class hierarchy to represent the concept of test programs.
- The tests will be run on the Java Compiler API, initially running the Java 5 compiler in a hardcoded fashion. User-selected compilers will be implemented in a later iteration. The tests will be run in a single execution, without the option of pausing the testing process.

The respective parts of the iteration were completed largely on time, including compatible interfaces being delivered to the central project. The narrow interfaces between the three separate areas of the project meant that connections and interfacing was the smallest stage in the iteration. However, the final running prototype of the first iteration failed to be delivered on time, due to a few small factors delaying the combining of the GUI with the other code units. Some of these are listed below, along with other issues.

Problems The GUI ran into problems when it needed to store data in an ordered fashion. The lack of dynamic arrays in Javascript meant that it was difficult to represent an ordered list of items, and so the GUI team implemented their own version of a Map, allowing them to apply an ordering to elements of a Test Program, which was important not only for the structure of the visual representation, but also for the data sent to the Generator.

The GUI team also had problems converting Javascript objects into Java *beans*, which we hadn't foreseen in planning stages. The conversion of JSON objects is 'shallow' in the sense that if the object contains a collection of JSON objects, these nested objects are merely cast as *DynaBeans*, whose type is not

recoverable. A post-processor was implemented that currently solves this problem.

The *Generation* code was completed with no hindrances, however some refactoring was performed when it was realised that a single core *Generator* class would be unwieldy. More extensible designs were considered, but in keeping with our XP practices, a simpler, less extensible solution was sought, with an aim to refactor again later in the development process, if extensibility becomes more desirable. The current solution to the single class unit was to make several *Generator* classes, all specialised, sharing core functions in a single class.

The *Compiler* unit was created with few problems, although research into future developments has shown that the work is considerably less straightforward than first thought. In particular, loading nonstandard Java compilers using the Sun Java Compiler API is more difficult than we had hoped it would be. Initial solutions have been found for the upcoming iterations, using command-line programs.

Overall, despite a delayed release, all the core code that we had hoped to write was written by the deadline. The changes to code structure did not affect the overall success of the iteration.

Review of Report One

Requirements & Extensions Overall, the team is making excellent progress along its plan outlined in the previous report, unusual though this is. Because of this, as well as great support from our supervisor, the requirements for the project remain unchanged. It's worth noting, however, that we aim to investigate making the *Generation* code extensible, allowing future users to use an API-like interface for extending our code to generate more interesting test cases. This has not been added as an official requirement, but the team will investigate it as a possible extension.

Development Methods Our development methods are serving us well on the whole, however the team is not meeting as often as they would like, and this has resulted in a few issues of communication between individual units. Notably, miscommunication over the interface between the GUI and the *Generation* code led to some duplicate code being written on both sides. This has been resolved, but the team must meet more frequently in future to discuss development directions and update each other on the progress of the work.

Iteration Layout The team is making very good progress through its iterations, to the point where some work is being done far ahead of time. At the time of writing this report, the second iteration is drawing to a close, but some features in the GUI from later iterations are already being considered and partially implemented. The iterations will be adjusted to reflect this. The original report shows the fourth iteration as including support for data storage, but it is now likely that this will be implemented in the third iteration.

Other notable changes include the rescheduling of user-specified *Compiler* uploading into the final iteration. This is largely because the *Compiler* work for the fourth iteration is heavily dependent on this feature, and our research indicates that preparing to implement the user upload feature is harder than its

actual implementation. Iteration three will allow for some time for this to be planned in more depth before attempting to integrate it. Iteration four will also allow time for the Generator team to investigate refactoring for extensibility. Again, iteration three will allow for some planning.

Future Iterations The revised iteration plan is below. Note that most iteration deadlines have been pushed back three days to their following Monday, to allow a weekend for interfacing. This is a minor alteration done as a result of problems met during the first iteration cycle.

Iteration 2 Estimated Completion Date - Monday 17th November. This iteration will add support for Fields and Method Headers to the generation of programs, and add according GUI options. The Java Compiler section should now allow for the user to upload their own compiler, and the GUI will express the results in a more intelligible format.

During this stage, testing of the state space coverage will begin using pre-calculated tests.

Iteration 3 Estimated Completion Date - Monday 1st December. This iteration will add support for Method Bodies to the generation of programs, with refactoring of earlier generation code if needed. The GUI will be extended to match this. Configuration options should now be complete by this stage, and the GUI will be able to estimate the size of the test program the user is requesting, as well as pause the execution of the tests on the compiler.

Iteration 4 Estimated Completion Date - Friday 12th December. This iteration will finalise support for Method Bodies, and optimise the generation code, updating the estimations in the GUI. The Java Compiler now compares a user-supplied compiler with the oracle compiler, and the GUI analyses the results to present a detailed summary of the two output streams. Parallelisation may now be implemented in the compiler execution stages. Testing can now be saved once it is paused, and resumed at another time.