University of Zurich
Department of Informatics

Software Quality Exercise 2

Testing and Debugging

1 Information

1.1 Dates

- Release: 12.04.2010 2pm
- Deadline: 26.04.2010 2pm
- Discussion: 10.05.2010 3.30pm

1.2 Formalities

Please submit your solution as a *pdf* and submit it via email to `jeanneret@ifi.uzh.ch`. The subject of the email must begin with *[FS 10 SWQ]*. Exercises can be solved and handed in in groups of two. Every member of a group must be able to answer questions about the group's solution. The document must include the names of group members. Finally, assignments are written in English, but feel free to write your answers in German (or in French) if you like to do so.

2 Introducing ImageJ

ImageJ is an image processing tool written in Java. With the help of plugins and macros, it can be extended with new features, such as the support of new file formats or new analysis. Still, the current architecture of ImageJ has reached its limits in terms of extensibility: many requested features (such as the support of dynamic charts or the ability to use ImageJ on a cluster of computers) cannot be implemented unless the tool is deeply refactored.

- The source code of ImageJ is hosted on the SVN repository located on *Daiquiri*. In previous exercise (3.2), you already checked out a local copy of this repository. A simple update (`svn up`) is enough to download the source code.

- For this exercise, you will need JUnit. This dependency has been mentioned in the `pom.xml` file of the project. Therefore, Maven2 will download it for you (when executing `mvn install`).
- In the repository, there is no metadata about the project for any IDE. Nevertheless, Netbeans can import Maven2 projects while Maven can create an Eclipse project (with the command `mvn eclipse:eclipse`). Note that before importing such a project in Eclipse, you need to setup the classpath variable M2_REPO (Window, Preferences, Java, Build Path, Classpath Variables) to the repository of Maven2 (which is usually located in `$home/.m2/repository`, where $home is your home directory).
- To execute ImageJ, you can either launch it within Eclipse (the main class is `ImageJ`) or use the *jar* file built by Maven2 (`java -jar target/ImageJ-1.4.3q.jar`).

One of the problem of ImageJ's current architecture is that ImageJ data models and image processing algorithms are tightly coupled to the GUI. Find 2 examples in the code where a single object both transforms images and displays something in the GUI. Discuss, in maximum 10 sentences, why this problem impacts both the extensibility and the testability of ImageJ.

3 Testing ImageJ

In Trac, there are 10 testing tasks available. Pick one of them (1 per group) and mark it as accepted (so that no assignment is solved twice). Prepare a test plan for your assignment and explain what kind of test it is and how you have chosen your test cases (you are free to choose your coverage criteria).

Automate your tests with JUnit 4[1]. For this purpose, create a new class in the `src/test/java` directory, so that your tests are run by Maven2 during the build. Before checking in your contribution, take note of the following points:

- If you need to use image files, you can place them in the `/src/test/resources` directory.
- You may have to modify ImageJ to perform your tests. If you do so, make your changes in such a way that it preserves the behavior of ImageJ from the viewpoint of an user.
- System tests can be implemented at the presentation or function level. Your tests can open windows and dialogs, but make sure that your tests close them programmatically, otherwise the continuous build may not terminate.
- Make sure that the project can be built correctly (tests can fail though, you are not supposed to repair the defects your tests have uncovered) before checking in your modifications. After the commit, verify that Hudson has built the project. If the build fails on the build server, it is very likely that it cannot be built by your colleagues either.
- Do not forget to include additional files in your commit (with the command `svn add`).
- In the commit's comment, refer to the Trac ticket. Once you have completed your task, close the ticket
- Report all related commits on the ticket.

Which difficulties have you encountered when automating your tests?

[1]See `http://daiquiri.ifi.uzh.ch/trac/swq10/wiki/JUnit` for some documentation on JUnit 4.

4 Improving the source code of ImageJ

In Trac, there are 10 "enhancement" tasks available. Pick one of them (1 per group) and mark it as accepted (so that no assignment is solved twice). Each assignment concerns one class. Assess the quality of its source code according to what you have learned in the Software Engineering lecture (chapter 6). Fix some of its issues (at least three) and check-in your modifications. For example, you can rename variables or methods, break methods into smaller one, document methods or introduce enumeration types. Many IDEs, including Netbeans and Eclipse, provide tool support for refactoring Java programs. Your modifications must preserve the behavior of ImageJ. Report all related commits on the ticket before closing it.

Once you have accomplished both the testing and improvement assignments, create a snapshot of the ImageJ project in the repository. The name of your tag must contain your Trac usernames.

5 Static Analysis

Static analysis of source code can help to hypothesize about the localization of a defect. Find data and control dependencies in the `stats` method of the `StaticAnalysis` class (within the ImageJ project). Chapter 11, from the Software Engineering lecture, may reveal helpful for this task.

a) What are the Set-Use, Use-Set, Set-Set relationships in this program?
b) Represent these relationships in a Program Dependency Graph (PDG).
c) Compute the static forward slice for
   `boolean inword = false` (line 22).
   Mark the corresponding path in the PDG.
d) Compute the static backward slice for
   `System.out.println("Number of words:  " + nw);` (line 38).
   Mark the corresponding path in the PDG.
e) How can you find dead code (code that is never executed) in a PDG? What other problems can be discovered with PDGs?

6 Hypothesizing about a Defect

On the website of the exercises, you can find a Java program called ArabianToRoman.jar. This program translates positive integers (smaller than 4'000) to the corresponding roman numeral. For example, 21 will be translated to XXI. You can launch this program with the following command:

```
java -jar ArabianToRoman.jar 21
```

It turns out that this program does not translate the number 16 correctly. Follow the scientific method (chapter 5, slide # 22) to come up with the best diagnostic (which inputs produce an error and what may be its cause) as possible. Report the complete history of your diagnostic in a table containing (a) your hypothesis, (b) your test cases and (c) the result of your test cases.