Steven Miller - stevem62
Andrew Tran - atran35

Project focus: Test Generation - tracking code coverage

Motivation

Randoop is a test generation tool which takes as input a program's source code, and outputs JUnit tests. It randomly selects methods to test, and provides random inputs to those methods. Programmers writing their own tests may be biased to write tests that cover only the cases they have considered, which can have anywhere from minor to severe consequences. Randomization may expose cases that have not been considered, but we think we can do better.

One metric that can provide greater confidence in testing a programs "correctness" is branch coverage. By branch coverage, we mean testing the different conditional branches within the code. If Randoop were to take branch coverage into consideration, it could potentially find more bugs. Passing tests would also give greater assurance to developers that their code meets specification requirements. While it is obvious that 100% branch coverage (if even achievable) does not ensure a bullet-proof program, creating tests to target sections of code is arguably better than rolling the dice every time and potentially testing the same input domains.

If we were able to integrate this into Randoop, it would be of great use to software developers. Testing is not something the average software developer enjoys, but is completely necessary. It is in everyone's best interest if unit tests can expose bugs, and including branch coverage as a metric is a great way to do that. Not to mention it can save a lot of money!

Approach

The high level approach is that our code would parse some input source code into form that we can work with in Java (probably an object), and detect conditional branches within the source code. Once the branches have been identified, our program will try to pick inputs which will lead to the different branches. These inputs will be used in Randoop for generating the test(s) for the given method. Note that multiple tests may be required to reach a particular threshold for branch coverage.

The main difference between our approach and what Randoop is doing right now is that we plan to choose arguments within specific domains instead of generating them at random.

One limitation of this approach is that if your code is written in such a way that a branch is never reached, you may not reach the threshold for branch coverage. A good example of this is making a conditional branch that isn't going to be reached now, but is written in preparation for future code that will need that condition.

Challenges and risks

The biggest challenges we see in developing the product on schedule would be the ramp up time required for getting familiar with Randoop, and properly integrating our code into the project. We will

have to be in contact with developers who are currently working on, or have worked on Randoop when we have questions that we cannot find answers to in the documentation or other online sources.

When it is done, if we compare the amount of code covered by the tests between the old Random implementation and the new Coverage implementation, the new one would cover more code.

When our project is complete, we can measure success by comparing the amount of branch coverage of a code base using the current Randoop implementation versus the branch coverage using Randoop with our added functionality.