Color of Benjamin
Color of Jesper

Introduction

Ever dreaded the thought of writing tests? Ever felt "why should I write tests? My code ALWAYS works!". Do not fret, friend, for we have the answer! EvoSuite! The program that writes tests for you! And even better, all those pesky metrics your boss is bothering you about? EvoSuite will optimise those for you, getting him off your back for all future! Next slide!

Slight theatrical pause

intro - same page cov - understand sbp
But enough of the infomercial, what is EvoSuite really? Next slide!I am Jesper,and I am Benjamin.We will try to explain automatic test creation for you with EvoSuite. To actually understand this, we need to first be on the same page about Coverage metrics, and understand search based programming. Next slide!

Coverage Metrics

many meterics - id bugs - same page
So most of you have probably heard or used some sort of coverage metric, whether it be in the software engineering fundamentals course or elsewhere. There are many of them, all specifying different things, but their point is to try to identify bugs in your code. To understand EvoSuite we need to be on the same page regarding the metrics that EvoSuite measures.

Line coverage

Simple, drawbacks, for loop, conditional operator
The most simple of metrics really, how many of all lines of code are executed by the tests? A high percentage score means that most of the program is executed by at least one test. However, this definitely has it's drawbacks. For example, a simple for loop will be executed if we run it once, but what if there are effects only if we run it twice? Line coverage can not necessarily say whether this is covered or not. Same thing can be experienced if we use a conditional operator in place of a normal if else block.

Branch coverage

opt. cov branches, outcomes for decsions
In contrast branch coverage is the percentage of branches in the code covered by the tests. One optimises it essentially by trying to run every bit of reachable code at least once during the tests. The main point here is that every decision point of the program should have all of its possible outcomes covered.

Mutation coverage

Bad Bad Joke
I think you mean corny joke ;p

What do you call a test suite that maximises mutation coverage? An evolutionary suite! (EvoSuite, get it?)

Actual script

Complicated, introduces mutants, detect. removing line, + → -
Mutation testing is perhaps the most complicated, and maybe also the most useful of the different coverage techniques. The technique relies on the creation of so called mutants, altered versions of the program, or function, under test that should behave differently. The aim of the test suite is then to detect the mutant, that is, fail if a mutant is introduced into the code base. The score is then calculated based on the number of mutants found by the test suite. Examples of mutants are for example removing a single line of a function, exchanging some operator or similar.

Next slide!

Search based programming

inteperet problems
Another important technique for EvoSuite is *Search Based Programming*. You might be familiar with linear programming or dynamic programming. Just as many problems can be reinterpreted as dynamic problems, can many problems can be interpreted as a optimization problems that are suited for search based algorithms. Next slide!

ACO, TSP, Scheduling
Such algorithms use metaheuristic search techniques such as simulated annealing or ACO to solve software engineering problems such TSP or scheduling.

approx max, trad slow
Search based programming may be used to approximate the maximum solution, where traditional algorithms could produce an exact solutions, but where the time to find such solution would be far too great to be practical. Next slide!

Relation to EvoSuite

point at screen
EvoSuite uses a search based techniques, namely an evolutionary algorithm to generate test suites. [1] It starts off with some set of randomly generated test suites. These test suites are evaluated against some or all of the coverage metrics mentioned above. The best ones are selected to reproduce. Reproduction here is done by exchanging some randomly selected set of test between two test suites. Finally, some mutations to the individual tests might be introduced. A mutation could for example be adding or removing a line. The new set of test suites are then

evaluated again. This continues until some test suites is complete or the allotted time is spent. Next slide!

Example and applications of EvoSuite

Readable, expect incomprehensible, single assertion

Behind me you see a couple of examples of tests generated by EvoSuite on a simple power algorithm. Also noteworthy of these tests are that they are remarkably readable. One would expect computer generated code to be incomprehensible, but since it simply puts a single assertion in each test here, it is not difficult to understand the purpose of each test.

Not equal, gems

Not all tests generated by EvoSuite are as sensible as these however. In our testing we encountered a few gems, like this one. Next slide!

From the tests you can clearly see accepted input of the function, but we also discovered some potential bugs. Next slide!Or isn't 10^{-2237} equivalent to $1 \ldots$? Clearly something is wrong in the code, which this generated test shows.

applications

Lazy, applications, unexpected behaviour, Legendre, ease of testing, simple, bugfree

While I may have joked previously about EvoSuite enabling me to be lazy since my code is, obviously, always correct (case in point). EvoSuite does have real applications that are not simply laziness. Looking at the above example, we see that there is clearly unexpected behaviour in our code. Of course, $10^{-2237} \neq 1$. So EvoSuite can certainly be a tool for discovering actual bugs for cases you did not yourself consider. One may of course ask, considering the function we chose, if you are expected to implement a power function making use of Legendre polynomials for the power function with reel numbers, and how testable it should be, but as long as only integers are involved, especially in the exponent, one would assume we should be able to create a good enough one ourselves that should be easily testable.

Next slide!

Conclusion

cool tool - areas - semantic errors

EvoSuite is a really cool tool, that certainly can be used for improving test suites and finding bugs. Writing tests is hard and EvoSuite can help finding places where your test suite is lacking or finding semantic errors.

complement

You shouldn't be putting all your eggs in one basket however. As we saw can EvoSuite sometimes generate some interesting tests. But it can certainly be used as a complement.

In short, EvoSuite really shows that Unit tests is not only about writing code.

Referenser

[1] Gordon Fraser and Andrea Arcuri. Evosuite: automatic test suite generation for object-oriented software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 416–419. ACM, 2011.