Software Testing, Quality Assurance and Maintenance Winter 2019

Lecture 17 — February 27, 2019

Patrick Lam *version 1*

We've talked about mutation testing in the past few lectures. I thought I'd summarize some recent research out of Waterloo, in collaboration with the University of Washington and the University of Sheffield, about: (1) the effectiveness of mutation testing; and (2) what coverage gets you in terms of test suite effectiveness.

Is Mutation Testing Any Good?

We've talked about mutation testing as a metric for evaluating test suites and making sure that test suites exercise the system under test sufficiently. The problem with metrics is that they can be gamed, or that they might measure not quite the right thing. When using metrics, it's critical to keep in mind what the right thing is. In this case, the right thing is the fault detection power of a test suite.

Some researchers set out to determine just that. They carried out a study, using realistic code, where they isolated a number of bugs, and evaluated whether or not there exists a correlation between real fault detection and mutant detection.

Summary. The answer is **yes**: test suites that kill more mutants are also better at finding real bugs. The researchers also investigated when mutation testing fell short—they enumerated types of bugs that mutation testing, as currently practiced, would not detect.

Methodology. The authors used 5 open-source projects. They isolated a total of 357 reproducible faults in these projects using the projects' bug reporting systems and source control repositories. They they generated 230,000 mutants using the Major mutation framework and investigated the ability of both developer-written test suites and automatically-generated test suites (EvoSuite, Randoop, JCrasher) to detect the 357 faults.

For each fault, the authors started with a developer-written test suite T_{bug} that did not detect the fault. Then, using the source repository, they extracted a developer-written test that detects the fault. Call this suite T_{fix}. Does T_{fix} detect more mutants than T_{bug}? If so, then we can conclude that the mutant behaves like a bug.

Results. The authors found that Major-generated mutation tests could detect 73% of the faults. In other words, for 73% of faults, some mutant will be killed by a test that also detects the fault. Increasing mutation coverage thus also increases the likelihood of finding faults.

The analogous numbers for branch coverage and statement coverage are, respectively, 50% and 40%. Specifically: the 357 tests that find faults only increase branch coverage 50% of the time, and they only increase statement coverage 40% of the time. So: improving your test suite often

doesn't get rewarded with a better statement coverage score, and half the time doesn't result in a better branch coverage score. Conversely, improving statement coverage doesn't help find more bugs because you're already reaching the fault, but you aren't sensitive to the erroneous state.

The authors also looked at the 27% of remaining faults that are not found by mutants. For 10% of these, better mutation operators could have helped. The remaining 17% were not suitable for mutation testing: they were fixed by e.g. algorithmic improvements or code deletion.

Reference. René Just, Darioush Jalali, Laura Inozemtseva, Michael D. Ernst, Reid Holmes, and Gordon Fraser. "Are Mutants a Valid Substitute for Real Faults in Software Testing?" In *Foundations of Software Engineering* 2014. pp654–665. `http://www.linozemtseva.com/research/2014/fse/mutant_validity/`

What Does (Graph) Coverage Buy You?

We've talked about graph coverage, notably statement coverage (node coverage) and branch coverage (edge coverage). They're popular because they are easy to compute. But, are they any good? Reid Holmes (a former Waterloo CS prof) and his student Laura Inozemtseva set out to answer that question.

Answer. Coverage does not correlate with high quality when it comes to test suites. Specifically: test suites that are larger are better because they are larger, not because they have higher coverage.

Methodology. The authors picked 5 large programs and created test suites for these programs by taking random subsets of the developer-written test suites. They measured coverage and they measured effectiveness (defined as % mutants detected; we've seen that detecting mutants is good, above).

Result. In more technical terms: after controlling for suite size, coverage is not strongly correlated with effectiveness.

Furthermore, stronger coverage (e.g. branch vs statement, logic vs branch) doesn't buy you better test suites.

Discussion. So why are we making you learn about coverage? Well, it's what's out there, so you should know about it. But be aware of its limitations.

Plus: if you are not covering some program element, then you obviously get no information about the behaviour of that element. Low coverage is bad. But high coverage is not necessarily good.

Reference. Laura Inozemtseva and Reid Holmes. "Coverage is Not Strongly Correlated with Test Suite Effectiveness." In *International Conference on Software Engineering* 2014. pp435–445. `http://www.linozemtseva.com/research/2014/icse/coverage/`