Response Paper

Use of Relative Code Churn Measures to Predict System Defect Density (Nagappan et al.)
Coverage Is Not Strongly Correlated with Test Suite Effectiveness (Inozemtseva et al.)
Sufficient Mutation Operators for Measuring Test Effectiveness (Namin et al.)

Rafael Kallis

March 11, 2017

One of the biggest challenges in software engineering is measuring the number of failures a piece of software has before it has been completed and shipped to the customer. The papers try tackling the problem by encouraging, or not, some methods and metrics which can help achieve a feasible solution. As identified by Nagappan et al. relative code churn is an efficiently calculated metric which approximates defect density quite good. Code churn also proves as an efficient metric to discriminate faulty and non-faulty binaries.

Inozemtseva et al. on the other hand, present results which do not prove, nor disprove, that code coverage is strongly correlated with test effectiveness. Based on their findings they also conclude, that using a fixed coverage value as a quality gate doesn't imply having an effective test suite. I completely agree with their finding and from my personal experience, it is quite easy to "cheat" the coverage metric, implying that code coverage doesn't necessarily improve code quality. Additionally they mention another metric, the mutation score, a potential substitute with a strong relationship with test effectiveness. Other papers have expressed the same idea.

Mutation analysis is a methodology which generates possibly faulty variants of programs, so called mutants, and feeds them to the test suit. Simply put, mutation testing acts as a test for test suites. The mutation score, or adequacy score, is then determined by the number of mutants exposed by a test case. Namin et al. analyzed and proposed a solution to a common problem in mutation testing, expense of calculation. Namin et al. try to discover a sufficient set of mutation operations which have the lowest total cost, in contrast to other papers which tried to find a set with lowest number of mutations. The set of operations also need to have a strong relationship with the actual adequacy score. They discovered a model which only contains 7.4% of the mutants generated by the Proteum system, the most comprehensive mutant generation system at the moment, and additionally predicts with very high precision the actual adequacy score.

All discoveries are in my opinion quite significant, and a combination of their findings could contribute to an increase in code quality when used correctly. In my eyes, the combination of code churn measures and mutation testing could leap software development forward. Using the code churn to efficiently approximate defect density and if exceeding a fixed defect density tolerance, using the more computationally intensive mutation analysis to improve existing, or generate, new tests in order to improve test effectiveness and keep defect density controlled could revolutionize software testing.