Development and Analysis of Mutation Testing

Joe Cummings

March 14, 2019

1 Hints on Test Data Selection: Help for the Practicing Programmer

```
@article{HintsOnTestDataSelection,
author={R. A. {DeMillo} and R. J. {Lipton} and F. G. {Sayward}},
journal={Computer},
title={Hints on Test Data Selection: Help for the Practicing Programmer},
year={1978},
volume={11},
number={4},
pages={34-41},
doi={10.1109/C-M.1978.218136},
ISSN={0018-9162},
month={April},
}
```

Summary

In this paper, Demillo et. al. propose mutation testing as an effective method to evaluate the health of test suites and briefly explore the rationale on why it might work. First, they conjecture that many cases tests of a program that uncover simple errors are also effective in uncovering much more complex errors. If this so-called Coupling Effect is true, it can be used to save work during the testing process. Mutation takes advantage of this hypothesis by making single syntactical changes to the source code of a project, called a mutant. If a mutant passes the test cases the programmer knows that he or she needs to add new tests to differentiate between the correct program and the mutant. This process continues until the test suite can detect all mutants, in which case the test suite is determined to be adequate.

Evaluation

This is the development on Lipton's ideas as proposed in his student paper, *Fault Diagnosis of Computer Systems* and the first time we see a full mutation testing system (based on Fortran). It also touches on the underlying theories, the Competent Programmer Hypothesis

and Coupling Effect, that are still being investigated today. This paper started a field of research that has only been increasing in relevance.

2 Mutation Analysis

```
@article{MutationAnalysis,
author = {Acree, A. and Budd, T. and Demillo, R. and Lipton, R. and Sayward, F.},
year = {1979},
month = {09},
pages = {92},
title = {Mutation Analysis}
}
```

Summary

In this paper, Acree proposes a more abstract framework to understand testing a program. In it, an original program, P, is said to be correct if and only if $\forall x P(x) = f(x)$ where x is an input to the program and $f(x)$ is the test oracle. This is an intractable set of inputs, so Acree proposes a proxy that P is correct if and only if $P(x) = f(x)$ for a reliable test set T, which encompasses a smaller set of x. A test set is reliable if it is determined to be adequate according to mutation testing. Acree also empirically tests the efficiency of mutation testing and the validity of the supporting hypotheses.

Evaluation

This paper is an extension and expansion of Demillo et. al.'s work in 1978. The Competent Programmer Hypothesis and the Coupling Effect are brought under scrutiny as are the problems associated with the technique. It is a much more comprehensive and balanced work than the 1978 paper, informing of the pros and cons of Mutation Testing. We see here the individual topics within the space that will be focused on for the next 40 years.