Daniel Carton

Dr. Gregory Gay – CSCE 747

Coverage and Its Discontents

Coverage and Its Discontents seeks to shed light on two alternate hypothesis in software testing, Strong and Weak Coverage Hypothesis, or SCH and WCH respectively. SCH is trying to prove the correlation between code coverage a test suite has and the level of fault detection it will produce. WCH strives to fight that by claiming no matter what test suite or human efforts arise, there is little to no correlation between coverage and fault detection. For example, SCH suggests that by high coverage like highlighting every branch, every statement, and every input, that there will be great deal of success in the fault detection (or vice versa) no matter what test method is used. On the other hand, WCH is claiming no matter what the coverage is there will always be holes in coverage.

Frankly, SCH should be applied in real-world software regardless of what evidence does or does not support its assertion. SCH supporting evidence suggests a more obvious answer between the two. For example, this paper suggests that a "safe use for coverage is to examine missing coverage to identify 'holes'", but this requires the test suite should have covered *more* areas of code (12). Covering more areas of code of course points back to the strong coverage hypothesis. So yes I think this is an applicable hypothesis for the real-world.

The authors suggest that even if two test suites use the same branch and statement coverage, they might uncover different results. A classic example is the Heartbleed bug (10). This of course has to do with the methods that are being applied and why I would encourage multiple test suites to be used. By broadening coverage and, more importantly, *methods* we will prosper in fault uncovering. Similarly, using different human efforts also broaden fault discovery. I have seen this at work when different team members notice different problems with applications. It goes without saying that different people would go about user interface testing differently.

As mentioned earlier and easily assumed, greater coverage should uncover more faults, but there is a caveat. The assisting test oracles must be catching the bad behaviors. When a test is run in a line of code and misses data that is incorrectly altered, a good oracle must be there to catch it afterwards. For example at my work place the web developers are expected to validate user input. Even if it passes, it might not pass the C# controllers that ultimately handle the between stages of data and a SQL database. With this in mind I propose a team working on an oracle, and another working on various test suites and their coverages. Daily stand-up's, like in Scrum, will promote communication between the two to ensure the data is being checked twice. However this does suggest that twice the computations will be present.

To further these approaches, mutation testing can give great insight on which hypothesis to support. Mutation testing creates small changes in code in hopes that the original test will still catch the poor behavior. Another positive perspective of mutation analysis is that the vast number of mutations don't all have to be considered because of manual exemption. Furthermore

proving that a test suite does in fact catch all the mutations would support evidence that coverage does or does not correlate with fault detection, because there is now coverage where there wasn't before (4).

More research needs to be done in the extreme cases of coverage. Think about an exponential graph that takes a long time to slope upwards. With the x-axis being coverage and the y-axis being fault detection. Is it possible that a test suite could take a relatively long time to prosper results? These might explain the outliers in figures one and two.

Lastly I think there should be a Diverse Coverage Hypothesis – By testing with multiple test suites, with coverage on a random basis, then the correlation between coverage and fault detection *might* be high. This could potentially be a good research topic and the results would be interesting.