Visualize the Effect of Test Suite and Fault Characteristics on Coverage and Detection of Faults

Bryan (Dung) Ta
Dept. of Computer Science
University of Maryland, College Park
bryanta@cs.umd.edu

Chenglin Chen
Dept. of Computer Science
University of Maryland, College Park
chenglin.chen@gmail.com

I. INTRODUCTION

As new software-testing techniques are continuously developed, developers have more choices yet have more difficulty in selecting an effective one. As a consequence, the effectiveness of software testing techniques at detecting defects must be evaluated. The most common evaluation method is empirical studies, in which one or more techniques are tried out on software with known defects. However, depending on the susceptibility of the defects to detection, the selection of the defects to use can affect the performance of the techniques. This issue leads to the need of studying the defect characteristics to uncover important insight such as types of test suites that generally detect more faults, types of faults that are generally more susceptible to detection, and types of test suites that detect certain types of faults better than the others.

Statistical approaches have been employed in [5], [4] to discover such information. However, in this project, we employ a different approach - Visualization with the supports of useful tools such as Spotfire [1] and HCE [2]. In comparison to the statistical approaches, Visualization highlights three main significant insights presented in section III, IV and V in more expressive and interactive way, yet less expensive in terms of time and computation. These insights not only help to solve the above issue in evaluating testing techniques, but also provide developers with useful information in test case design.

We acknowledge that the project should be conducted on many datasets to avoid threats to validity. However, due to the time constraints, we work on only one data set which is described in the next section.

II. DATA DESCRIPTION

The data set [3] is the GUI testing results of CrosswordSage 0.3.5 - a crossword-design tool. It includes 1185 data points, each in the format of <test suite characteristics vector, fault characteristics vector, Cov, Det>. There are 22 attributes in the data set. The first 8 ones (i.e. start with T.) are test suite characteristics, and the next 12 ones (i.e. start with F.) are fault characteristics, and the last two are Cov and Det which have values of 1 if the test suite covered/detected the fault and, of 0 otherwise. Figure 1 provides a description of all the attributes.

![Fig. 1. Dataset description](image-url)
III. INSIGHT 1: CLASS-LEVEL MUTATION FAULTS ARE MORE LIKELY TO BE COVERED BUT LESS LIKELY TO BE DETECTED THAN METHOD-LEVEL MUTATION FAULTS.

![Coverage and Detection comparison between class-level and method-level mutation faults](image)

Figure 2 contains two line charts showing that the fault characteristics of method of creation have opposing effects on fault coverage and fault detection. In coverage comparison, the blue line (class-level mutation faults) has a steeper slope than the red line (method-level mutation faults) which means a bigger portion of class-level mutation faults is covered by the test suites. In detection comparison, the order of the slopes is reversed, the method-level mutation faults (represented by red line) are more likely to be detected.

There is another way to visualize this. Figure 3 are bar charts for class-level mutation faults and method-level mutation faults respectively. In both graphs, the first bar represents the number of faults not covered (thus not detected) by the test suits. The second bar has two parts. The top part represents the number of faults being covered and detected and the bottom part represents the number of faults being covered but not detected. These two bar charts clearly show that class-level mutation faults have a higher rate of being covered but a lower rate of being detected by the test suites than method-level mutation faults.

![Fault coverage and detection for class-level mutation faults and for method-level mutation faults](image)

IV. INSIGHT 2: INTERACTION BETWEEN FAULT AND TEST-SUITE CHARACTERISTICS SIGNIFICANTLY AFFECTS FAULT DETECTION

The three pie charts in figure 4 show how different settings of T.Pair and F.CovBef change the ratio of faults being detected. The second pie chart in figure 4 shows that when T.Pair and F.CovBef are at the middle of their range values \(<T.pair = 0.27, F.CovBef = 0.11>\), the ratio between the number of detected faults and that of undetected faults are equal 50% :50%. However, when we changed the values of T.Pair and F.CovBef, we noticed an interesting fluctuation of the ratio. Specifically, the ratio increases in favor of detected faults when both T.Pairs and F.CovBef increase (i.e. 83.3%:16.7% in the third pie chart with \(<T.pair = 0.28, F.CovBef = 0.16>\)), whereas it increases in favor of undetected faults when both T.Pairs and F.CovBef decrease (i.e. 28.7%:71.3% in the first pie chart with \(<T.pair = 0.22, F.CovBef = 0.08>\)). These visualizations lead us to one significant observation: Faults lying farther from the initial state (higher F.CovBef) were also better targeted by test suites with greater event-pair coverage (T.Pairs). This insight suggests GUI tester improve event-pair coverage of their test suites if they want to detect deeper faults in software under test more effectively.
V. INSIGHT 3: INTERACTION BETWEEN TEST-SUITE CHARACTERISTICS SIGNIFICANTLY AFFECTS FAULT DETECTION.

The color mosaic feature of HCE tool helps us gain another significant insight: the event-coverage metrics for proportion of coverage (T.Pairs and T.Triples) affect fault detection without affecting fault coverage. This is shown via the visualization in figure 5 which highlights a higher frequency of color changes in Det than that in Cov. Obviously, the change between being undetected (0 - green color) and being detected (1- red color) in Det is much more often than the one between being uncovered (0 - green color) and being covered (1- red color) in Cov. This insight provides a useful guideline for test case generator for GUI-based software that: T.Pairs and T.Triples do not help to improve fault coverage.

VI. TOOL CRITIQUE

A. Spotfire

TIBCO Spotfire is a very nice tool that is comprehensive and easy to use. It has a wide selection of visualizations to help the users to explore their data, and its response time is very impressive. However, there is still room for improvement. For example:

- We would like to be able to use data from multiple tables in one graph.
- Sometimes the default setting of axis is confusing/misleading.
- There is no easy way to change the label and its font size of an axis.

B. HCE 3.5 test version

HCE is a very powerful tool to handle Multidimensional Data. It provides users with great controls over data analysis processes. We use the Color Mosaic feature in our work and like it a lot. One suggestion for possible improvement is: in Color Mosaic feature, it would be better to make the labels of the column names float so that users can still see the column names of the data when scrolling down the graph.

REFERENCES