CS639 – Program Analysis, Verification and Testing

Assignment-3

Spectrum Based Fault Localization

Name:- Govind Sharma

Program:- Mtech CSE

RollNo. :- 231110015

Spectrum-based Fault Localization (SFL) is a technique that uses test cases and code coverage information to estimate the likelihood of each program component being faulty.

Basically, the task is to determine whether a modified version of a program, referred to as P2, contains defects, and if so, to pinpoint the specific lines of code most likely to be at fault.

In this Assignment our main goal is to implement 3 functions which are FitnessScore, Suspiciousness and getRankList where we will compute fitnessscore by evaluating Uniqueness, Density and Diversity

Below is given how SFL works:-

1.Generate a collection of test-cases on the buggy program. This is called a test-suite.

2.Execute the test-suite and collect the execution outcomes. This is called an error vector.

3.Observe the test-suite and the outcome to determine which component is most likely to be buggy.

DDU metric is calculated for Fitness score and Optimize the structural properties of the activity matrix which make them ideal for diagnosing:

1. Density

2. Diversity

3. Uniqueness

Density:-

$$\rho = \frac{\sum_{i,j} A_{ij}}{MXN}$$

An ideal test suite has the maximum entropy = 1 Therefore, the ideal value of density is ρ = 0.5

Diversity:-

All test cases in a test suite must be unique. Same test cases are redundant and provide no extra information. To ensure that a test suite does not contain redundant test cases, maximize Gini-Simpson metric:

$$\delta = 1 - \frac{\sum_{k=1}^{l} |n_k|(|n_k|-1)}{N(N-1)}$$

Where n is the number of test cases having the same activity pattern. δ is the probability of two test patterns selected at random being of two different kinds. The ideal value of Diversity is δ = 1

Uniqueness:-

If two or more components exhibit the same activity pattern then they form an ambiguity group. Components within an ambiguity group are indistinguishable from one another. Therefore, we want the number of ambiguity groups to be as high as possible. Uniqueness is defined as:

$$\mu = \frac{|\Omega|}{M}$$

Ideal value of uniqueness is 1

DDU Metric :-

The quality of a test suite is quantified by:

$$DDU = \rho\prime . \delta . \mu$$

$$\rho\prime = 1 - |1 - 2\rho|$$

Optimal value of DDU is 1

In suspiciousness function we are calculating nothing but Ochiai Metric which is sus_score and ochiai metric is defined as:-

$$Ochiai(C) = \frac{C_f}{\sqrt{(C_f + N_f).(C_f + C_p)}}$$

Assumptions:-

- Program P is assumed to be a known correct version of the software. P2 is the version of the software that has undergone changes, and we want to determine if it contains bugs. It is assumed that P and P2 are both represented as a set of program components

- Test-suite T is generated using an evolutionary search-based test-suite generation framework in Kachua. The fitness function provided for test-suite generation aims to optimize the test-suite for fault localization.

- The test-suite T is represented as an activity matrix A, where rows represent test cases and columns represent components. The value in the matrix indicates the activity or execution status of a component for a specific test case.

- The buggy component in P2 is assumed to exist, and the goal is to locate it. The SBFL framework will rank the components based on their suspiciousness scores, and the buggy component is expected to have a high ranking.

Limitations:-

-The SBFL framework assumes that there is a single buggy component in P2. If there are multiple bugs in the code, the framework may not handle them effectively.

- The effectiveness of the SBFL metric in identifying the buggy component depends on the design of the metric and the nature of the code changes in P2. If the metric is not well-designed or the code changes are complex, the results may not accurately identify the buggy component.

- The efficiency and scalability of the SBFL framework are not discussed. For large codebases and extensive test-suites, the framework's performance may be a limiting factor.

Conclusion:

This assignment involves the application of Spectrum-based Fault Localization (SBFL) to identify potentially buggy components in a modified version of a program (P2). The process includes generating a test-suite, executing it on P2, and using SBFL metrics to rank the components by suspiciousness. The main assumption is the existence of a single buggy component in P2, and the effectiveness of the framework relies on the quality of the test-suite and the design of the SBFL metric. Limitations include the handling of multiple bugs, the effectiveness of the metric, and the scalability of the framework. Overall, SBFL is a useful technique for identifying potential code defects in regression testing scenarios.