A Novel Approach For Detecting And Correcting Errors Using Combinatorial Testing

R. Swethasree and B. Galeebathullah

1PG Student-CSE Sona College of Technology Salem, India.
2Assistant professor-CSE Sona College of Technology Salem, India.

Received 28 January 2017; Accepted 22 May 2017; Available online 28 May 2017

ABSTRACT

Combinatorial testing (CT) has been used to identifying fault interactions. The interactions which cause failure are identified at the earlier stage. The degree of interactions are identified still an open issue. Interactions are identified by applying data flow technique from the Decision to Decision (DD) path graph generated from the source code. Data flow fault interactions are identified and eliminated. Most of the testing works detect the errors for small sized system but our work is applicable for larger data. The testing process are used in large type of data so it take more time to analysis the errors so time complexity are high. Our purposed approach is reduced the time complexity.


INTRODUCTION

Software testing is a process of executing a program or application with intent to finding a software bug. Software testing is expensive and time consuming that leads to the software system. The Testing process is never ending process, i.e. it cannot be said that particular software is 100% error free. Therefore, we have to define when we start testing and when to stop testing. In order to maintain entire testing process it is always better to have a STLC (Software testing life cycle).

Combinatorial Testing (CT) is focus on testing combination of values only. All combination of values are testing exhaustive manner. Different types of testing are used to identify errors or faults, for ex: unit test and mutation testing are give the effective test cases that similarly used for combinatorial testing it is only focused on to identifying the faults, that faults are occurred due to the interaction of different parameters. CT task goes beyond testing individual variables. In CT task are verified using different combinations of variable are handled correctly in the systems.

There are two approaches used for CT:

- Testing of configuration parameter value.
- Testing of input parameter value.

Interaction failures are classified into two main categories: type one and type two interaction failures. Type one interaction failures occur when an interaction variables leads to an execution of a code in which a fault
exists. For example, a transaction may be fail only for a particular account type when the amount is less than a certain value. Type two interaction failures occur when an arithmetic computation leads to an incorrect result. Here, placing a more operator causes interaction among variables to produce an incorrect result. Thus, a technique is required to identify interaction faults.

Data flow analysis techniques only focus on the flow of data through the program. The flow of data is analysis with the help of usage of data. They are two usages defined are c-use (computation use) and p-use (predicate use). A variable is considered c-used it occurs in an output statement, as a parameter within a function. A variable is considered p-used it is used as a condition in a branch statement.

In this paper, proposed approach is identifies the interactions among variables that exist in the code. Data flow analysis is applied on the flow graph obtained from the source code that is used to identify the interactions. Since the focus of the approach is to identify interactions. The initial results is the approach is able to achieve a considerable reduction in the number of interactions to be tested, only the identified interactions are tested compare to all the possible interactions among variables for a given value of strength of coverage.

2. Related Work:

Combinatorial testing has become a well accepted testing method. A large number of research have focus on CT. Researchers are broadly classified them into eight categories [6]. Most of the researchers are focused on t-way testing that all are having algebraic method [7], simulated annealing [8], ant colony optimization, swarm optimization and Genetic Algorithm [10]. Not more research has been done in the area involving modeling of the input space. The input space involves identifying the basic elements such as parameter. The element take input is combinatorial model and can be identified from requirements documents, designs, codes, etc. Cheng et al. [11] constructed the graph for each system and used a greedy algorithm to reduce the count of input parameters. Satish et al. [13] proposed approach is to identify parameters and their values from UML activity diagrams. Schroeder and Korel [14] proposed approaches to reduce the number of test cases by identifying relationships between program input and output variables. It is assumed only a few input variables affect the value of an output variable. Thus, for each output and input variables are identified and test cases are generated. The union of test cases generated for each output variable gives the full complete test set.

3. Proposed Approach:

In this section the proposed approach that identifies interaction of variable from an object oriented program is discussed. The source code is transformed to a DD graph. A DD path graph is used, where each node represents a statement and each edge represents the flow of control of statements within the program. In the proposed approach, a variable is considered c-used if it is only occurs in a subscript expression or assignment statement. As the proposed approach identify the interactions. Then different strengths of interaction are identified.

1) The proposed approach first takes source code as input. The source code is transformed to a DD graph.

2) The variables are identified in the DD path graph. At each node in the DD, the p-use and c-use variable are identified.

3) The DD is converted to a RDD (Reduced Decision to Decision). The overall are graph are not modified a node is eliminated. If the number of path is same, the elimination is achieved by using certain rules that is given below.

   a) Rule 1–if the in-degree and the out-degree of a node are 1, then edge is drawn from fits predecessor node to its successor node and it’s deleted.

   b) Rule 2–if the in-degree of a node is 0, then out degree of the node is 1, the node started part is deleted from the graph.

   c) Rule 3 – if the in-degree of a node is 1 and the out-degree of the node is 0, then the node is end node and is deleted.

   d) Rule 4 – if the in-degree of a node is greater than 1, then the node is a merge node and cannot be removed.

   e) Rule 5 – if the out-degree of a node is greater than 1, then the node is a branch node and cannot be removed.

4) After reducing the graph to a DD, the nodes left in the graph are branch nodes.

   a) Identification of Type 1 interaction Failures.

   First the independent paths in the graph are identified. For each path, an empty set is created. Then each path is traversed to identify the interactions. While the path is traversing, if a variable is p-used at a node, then the variable is added to the set. When the traversal of the path, the set obtained contains all the variables that have been p-used at any node. For each path, the set of variable gives an interaction that may cause a type 1 interaction failure in the source code.

   b) Identification of Type 2 interaction failures.
The set of variables that are considered c-used at a node represents an interaction. The set of such interactions at any node in the graph gives the complete set of interactions that may cause type 2 interaction failures in the program.

The union of type 1 and type 2 interactions gives the set of interactions in the code. The resulting set of interactions identified may not be minimal.

**Fig. 1: Flowchart of combinatorial testing using Greedy Algorithm**

4. **Greedy Algorithm:**

   Genetic algorithm is a heuristic method based on survival of the fittest. It’s useful for when search space very large or too complex for analytic treatment. In iteration possible solutions or individuals represented as strings of numbers. All individuals in population evaluated by fitness function. The individuals allowed to reproduce, crossover, mutate function.

   In this all step are implemented to predict correct optimization value. It’s mainly used to reduce test case size. Greedy algorithm is used to reduce the test case size in RDD path graph that shows fig 6.
RESULT AND DISCUSSION

The input is given source code and some condition which is first loaded the combination is viewed, and genetic algorithms are applied to view the DD path graph. The graph only shows the combination of values and some rules are followed to reduce the graph.

Fig. 2: View combination

Fig. 3: source code

Fig. 4: DD path graph
Fig. 6: RDD path graph

If DD path graph are generated after the failure interactions are identified using dataflow technique then failure interaction are reduced.

Conclusion:

In this paper the problem of studied about the failure interaction. The failure interaction is mostly affecting the combination of values checking process. The existing system only used for structured program but our proposed system is implemented object oriented system. Further we will study about the time complexity.

REFERENCES