Testing approach for automatic test case generation and Optimization using GA

Mr. Sridhar Yemu1, Prof. Kapil Vhatkar2, Prof. Vipul Bag3

1 Nagesh Karajagi Orchid College of Engineering & Technology, Solapur
2 Nagesh Karajagi Orchid College of Engineering & Technology, Solapur
3 Nagesh Karajagi Orchid College of Engineering & Technology, Solapur

Abstract

Model based testing approach for both test case generation and test case optimization for object-oriented softwares using UML diagrams. The proposed approach addresses the issue of redundancy, size of test cases and optimization challenges. Automation of test case design process can result in significant reductions in time and effort, and at the same time it can help in improving reliability of the software through increased test coverage. The proposed approach uses genetic algorithm from which best test cases can be optimized. Moreover our method for test case generation inspires the developers to improve the design quality.

Keywords: Optimization, Genetic Algorithm, Reliability, UML

1. INTRODUCTION

Testing activities consist of designing test cases that are sequences of inputs, executing the program with test cases, and examining the results produced by this execution. Testing can be carried out earlier in the development process so that the developer will be able to find the inconsistencies and ambiguities in the specification and hence will be able to improve the specification before the program is written. It is still a major problem to meet the requirement specification for the systematic production of high-quality software. Many researchers are doing research on to find effective test cases to minimize time and cost. Hence, it is important to generate test cases based on design specifications. Unified Modeling Language has become the de facto standard for object-oriented modeling and design. It is widely accepted and used by industry. The complexity of system testing can possibly be attributed to the fact that it involves testing a fully integrated system that may be large and complex. Not surprisingly, system testing of typical systems often overwhelms manual test design efforts.

2. TESTING APPROACHES

Software testing is the process of exercising a program with well-designed input data with the intent of observing failures. Software testing addresses the problem of effectively finding the difference between expected behavior specified by the system models and the observed behavior of the implemented system. At present, software testing on the average makes up as much as 40% to 60% of the total development cost and would increase even further with rapidly increase size and complexity of software. As systems are getting larger and more complex, the time and effort required for testing are expected to increase even further. Therefore, automatic software testing has become an urgent practical necessity to reduce testing cost and time. Our research is focused on the use of UML models for the above said purpose i.e. automatic test case generation and test case optimization. We found so many researchers who have already worked in UML for test case generation [5]. Basic concepts on Object-Oriented Software Testing (OOST) strategy and use of UML diagrams as test model are covered. We also discuss some basic concepts of genetic algorithm (GA) and optimization. We gone through some research papers Those have covered extensively the various aspects of object-oriented software testing using UML diagrams. State diagrams or state chart diagrams are used to help the developer better understand any complex functionality or depict the dynamic behavior of the entire system, or a sub-system, or even a single object in a system. GA can be used to generate the test data using UML state chart diagram as described by Biswal [3]. Sometimes after coding developers don’t have time to test the software. Generating test cases from UML state chart diagram can solve this problem by generating them before the coding. Then the test cases can be generated as per the specifications of the software. Specifications can be in the form of UML diagrams, formal language specifications or natural language description. Software testing efficiency is optimized by identifying critical path clusters [8]. The test case scenarios are derived from activity diagram. The activity diagram is converted into CFG where each node represents an activity and the edges of the flow graph depict the control flow of the activities. It may be very tedious expensive and time consuming to achieve this goal due to various reasons. For example, there can exist infinite paths when a CFG has loops. GA has been applied in many optimization problems for generating test plans for functionality testing, feasible test cases and in many other areas. GA has also been used in model based test
case generation [3]. Various techniques have been proposed for generating test data/test cases automatically using GA in structural testing [2], [5]. GA has also been applied in the regression testing, object oriented unit testing as well as in the black box testing for the automatic generation of the test cases.

3. EXISTING SYSTEM

Several approaches to design test cases and application of Genetic Algorithm on software testing have been proposed by researchers. These approaches include generation of test cases from requirements specification i.e. black box testing or from code i.e. white box testing or from model-based specification [7]. Test case generation solely based on requirements specification completely ignores system implementation aspects. Further, it cannot reveal whether the software performed some tasks which are not specified in the requirement specifications. On the other hand, test case design from program code is cumbersome and difficult to automate. Besides, it also cannot reveal missing functionalities [2]. Further, the traditional system testing techniques black box as well as white box testing, achieves poor state coverage in many systems. The reason being that the system state information is very difficult to identify either from the requirement specifications or from the code [3]. an automated model-driven test case framework is therefore desirable. At the same time we have studied on optimization of generated test cases.

4. NECESSITY OF PROPOSED SYSTEM

1. Lacking in generalization and automation. These are two basic pit falls of present procedure to software testing. When it being automated it fails to generalize only for that particular application it fits.

2. The open problem is to generate an automatic with existing approaches for generalization.

5. IMPLEMENTATION

Algorithm Steps (Online Shopping)

- Construct the state chart diagram. (read from mdl file)
- Convert the state chart diagram into state chart graph \( G = (N, T) \)
- The will represent inputs
- The graph is traversed using DFS to select the functions (and identify their relational behaviour) for which test cases have to be generated and the input parameters and output values to be identified.
- The input values are identified (from their relational behaviour) using boundary analysis to generate various test cases
- nodes generated here will represent states
- Edges
- Test cases identified are converted to source code
- The Extended Finite State Machine (EFSM) from the code is generated using Model Unit model based testing mechanism.

- The output generated by the above tool is stored in a file which specify a refined set of test cases for the optimization

Test paths generated have to be optimized using genetic

1. Identify individual test paths
2. Assign weights to each function based on target nodes connected by the transition

More weight is assigned to a path which is more “critical”. Criticality of the path to test data generation is based on the fact that predicate (conditional), loop and branch nodes are given preference over sequential nodes during software testing.

3. Fitness Function

The fitness function is defined as:

\[
F = \sum_{i=1}^{n} w_i
\]

Where, \( w_i \) = weight assigned to i-th edge on the path under consideration

4. Probability

After all the fitness function values are calculated, the probability of selection \( p_j \) for each path \( j \), so that

\[
p_j = \frac{F_j}{\sum_i F_i}
\]

Figure 1. State diagram for Online Shopping

Figure 2. Generation of Test Scenario
6. RESULTS

1) To generate test cases are efficient and optimal.

2) The approach addresses testability, coverage criteria and automation issues in order to fully support system testing activities.

References


[6] Li Bao-Lin, Li Zhi-shu, Li Qing, Chen Yan Hong ,” Test Case automate Generation from UML Sequence diagram and OCL Expression”, International Conference on Computational Intelligence and Security 2007, pp 1048-52.
