FULLY AUTOMATED TEST CASE GENERATION IN SOFTWARE TESTING USING HILL CLIMBING ALGORITHM

S. Sharmila Jeyarani¹, Dr. R.S. Vetrivel²

¹Assistant Professor, Dept. of Computer Science, NPR Arts & Science College, Natham, Dindigul.
²Principal, NPR Arts & Science College, Natham, Dindigul.
¹sharmimca85@gmail.com, ²vetri76@gmail.com

ABSTRACT:

Software testing is set of activities conducted with the intention of finding bugs in software. Major part of testing the software involves the generation of test cases. A good test case should have the quality to wrap more features of test objective. By reducing the test case creation time we can improve productivity of software development, it also increases the productivity of software developers. The techniques for automatic generation of test cases try to efficiently find a small set of cases that allow an sufficiency criterion to be fulfilled, thus reducing the cost of software testing and resulting in more efficient testing of software products. The test cases are generated using the technique of boundary value analysis (BVA). It is the technique of making sure that performance of system is predictable for the input and output boundary conditions. This project demonstrates the generation of automated unit test case using Hill Climbing (HC) algorithm for real time scenarios like automated test case for dot net assembly. This project eliminates the time waste by creating the test cases automatically, it also enables easy maintainability and supports easy way to add, edit and remove test cases for sub sequent release of the software. This unit test case automation tool generates unit test cases and executes the test cases to provide the test execution result. This tool uses Microsoft .net reflection to re-factor the assembly and uses hill climbing algorithm to form the combination of test cases. It generates the boundary value test cases for all the methods available in each and every class in the input assembly. This tool supports NUnit and Microsoft test format.

Keywords: BVA, Hill Climbing Algorithm, Test Case Generation.
FULLY AUTOMATED TEST CASE GENERATION IN SOFTWARE TESTING USING HILL CLIMBING ALGORITHM

I. INTRODUCTION

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors. Automated Testing: Many programming groups are relying more and more on automated testing, especially groups that use test-driven development. There are many frameworks to write tests in, and continuous integration software will run tests automatically every time code is checked into a version control system.

While automation cannot reproduce everything that a human can do (and all the ways they think of doing it), it can be very useful for regression testing. However, it does require a well-developed test suite of testing scripts in order to be truly useful.

A unit is the smallest testable part of an application like functions, classes, procedures, interfaces. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. Unit tests are basically written and executed by software developers to make sure that code meets its design and requirements and behaves as expected. The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy.

Parameterized unit tests (PUTs) are tests that take parameters. Unlike traditional unit tests, which are usually closed methods, PUTs take any set of parameters. PUTs have been supported by TestNG, JUnit and various .NET test frameworks. Suitable parameters for the unit tests may be supplied manually or in some cases are automatically generated by the test framework. Testing tools QuickCheck exist to generate test inputs for PUTs.

This will generate a positive test case along with all feasible boundary value test cases. Microsoft .net reflector is used to re-factor the assemblies and structural based testing approach is used to generate the unit test cases. Feasible test case combinations are formed using the Hill climbing algorithms. This tool support NUnit and Microsoft testing format and creates test case project in two different formats. If a method has an object type parameter as input the object will be traversed to the root node till it reaches root node with predefined data type.

II. PROPOSED SYSTEM

Proposed System is a software tool which generates test cases for any assembly, this covers all feasible scenarios. This tool creates a positive test case along with all possible boundary value test cases, by which the behavior and structure of software unit is being verified. Multiple test case format like NUnit, Microsoft Unit testing is supported. By generating more test cases automatically with less amount of time, this tool helps to improve the built-in quality of the software. This tool can be converted as Microsoft Visual Studio Plug-in which makes the test case preparation more effective and efficient. By using the tool the time spent for test case preparation is saved, by which more time can spent in new developments and innovation ideas.

Whenever a new requirement is added, developer needs to just rerun the automated tool instead of creating new test cases and add it to the test case project. It will automatically describe the input data can be generated for test cases. Using .Net Reflection an assembly is parsed and processed by hill climbing algorithm to form combinations of test cases.

S. Sharmila Jeyarani And Dr. R. S. Vetrivel
NUnit and Microsoft test framework syntaces are used to create test case methods and projects.

**Advantages:**
- Reducing the cost of software testing
- Reducing human errors
- Increasing software products quality
- Reducing number of test cases
- Utilize the time in efficient manner
- It covers all possible test cases because it is automated
- Accuracy and Error free

### III. SYSTEM ARCHITECTURE

Software testing is the time and resource consuming task in the development cycle of the software. System Architecture for Automatic Test Case Generation is showed in Fig 3.1

Test case generation is important and critical task in software testing. In testing the tester is intended to create an optimal code of test data to test the program of the software. The quality of the software is determined by the number of error found by the test cases. It is difficult task to find the optimal case among the number of test cases.

Many efforts have been done by the researchers for solving the problem of optimization. They applied many search based algorithm for finding the optimal solution of test cases. The automation of the test cases is still an growing research area since the auto-mated testing tools simply generate test data with only less consideration on amount of time spent for test data generation and selection of test cases based on the test adequacy criterion.

### IV. AUTOMATIC TEST CASE GENERATION USING HILL CLIMBING

A wide range of different optimization and search techniques have been used. The most widely used methods are local search, simulated annealing, genetic algorithms and genetic programming. However, whatever may be the search technique employed, it is the fitness function that plays a major role and it captures a test objective and makes a contribution to the test adequacy criterion. Using the fitness function as a guide, the search seeks test inputs that maximize the achievement of this test objective.

Hill climbing (HC) is a local search algorithm. It needs that a neighborhood $N$ of the current sequence $S_i$ is defined. The search will move to a new solution $S_j \in N$ only if $S_j$ is better. If there are no better solutions in $N$, a local optimum has been reached. In such a case, a restart of the algorithm from a new random solution can be done.

Andrea Arcuri and Xin Yao [4] automatically generated test cases for structural testing of Java containers using Hill Climbing. A solution to the problem is a sequence $S_i$ of Function Calls (FCs) on an instance of the Container under Test (CuT). A Function Call (FC) can be seen as a triple: $< \text{object reference}; \text{function name}; \text{input list} >$. They used three
FULLY AUTOMATED TEST CASE GENERATION IN SOFTWARE TESTING USING HILL CLIMBING ALGORITHM

types of operations on Si for generating N: Removal of a single FC from Si, Insertion of a new FC in Si, and Modification of the parameters of a FC.

In general HC is a simple algorithm but may lead to a local optimum.

**Fitness Function**

Local search works best with a smooth landscape. But it gets trapped in local optima when the landscape is multi-modal. In this case, global search may perform better. A landscape with large plateau is very hard for all searching algorithms.

There are two types of search based methods.

- Local Search
- Global Search

The types of search based methods are shown in Fig 4.1

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle: first the developer writes an (initially failing) automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally re-factors the new code to acceptable standards.

**Boundary Value Analysis** [3], in this technique test cases are designed by taking the values of boundaries. The mainly focus of testing at boundaries rather than the centers values. In this technique errors are observed at the extreme values

V. **CLASSIFICATION OF TEST TYPE**

There are different types of software testing approaches. Respecting to the fact that test case generation is an important phase in software testing, test case generation approaches directly depends on the type of software testing. The following table represents a classification for different automatic test case generation approaches in terms of test type.

The quality of the software also improved by using this automated tool, this tool generates all feasible scenarios test cases with maximum accuracy, this lead to improve the testing and software quality. At present Test driven development [TDD] is followed by most of the companies in software industries, this automation test case generation tool helps the Test Driven Development [TDD] easy by quickly generating test case for each module in every phase.
This automation test case generation is done based on the structure and behavior based test case approaches.

### VI. CLASSIFICATION OF TCG APPROACHES IN TERMS OF ALGORITHM

<table>
<thead>
<tr>
<th>Approach</th>
<th>Idea</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structural-Based</td>
<td>The goal of structural testing is to cover test adequacy criteria.</td>
<td>1. Comprehensible 2. Simple Implementation</td>
<td>Errors discovered are very difficult to correct</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specification and complex to difficult understand</td>
</tr>
<tr>
<td>Functional-Based</td>
<td>The goal of functional testing is to test the functionality of the software under test.</td>
<td>1. Errors discovered are simply and not much costly to correct 2. Inconsistencies are found earlier</td>
<td></td>
</tr>
<tr>
<td>Gray-Box Based</td>
<td>Both structural and functional information are used for generating test cases.</td>
<td>It takes the benefits of both structural and functional based approaches</td>
<td>High Complexity</td>
</tr>
<tr>
<td>Non-Functional Based</td>
<td>Non-Functional testing is testing of “how” the system works.</td>
<td>Takes into account other behaviours despite of logical behaviours of system</td>
<td>Very complicated because it depends both hardware and software features</td>
</tr>
</tbody>
</table>

This automation test case generation is done based on the structure and behavior based test case approaches.
FULLY AUTOMATED TEST CASE GENERATION IN SOFTWARE TESTING USING HILL CLIMBING ALGORITHM

<table>
<thead>
<tr>
<th>Search Based</th>
<th>huge number of test cases</th>
<th>Large Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>In these approaches the problem of generating test cases are considered as an optimization problem, and try to find optimize solution</td>
<td>The results show the efficiencies of this approaches</td>
<td></td>
</tr>
</tbody>
</table>

| Data Mining Based | The main goal is analyzing the input/outputs of a program under test to reduce the number of test cases by eliminating unimportant and infeasible test cases | Reduces number of test cases results directly in the saving of software testing resources | 1. High complexity |

This Project uses structural based test type and uses search based algorithm. This test type is comprehensible and implementation also simple. Applications of the structural based test type are automatic test case generation for functions in procedural programs and automatic test case generation for some important classes in object oriented programs.

**CONCLUSION**

By creating test cases automatically, this automated test case generation tool eliminated the time waste and improves the productivity. This tool also improves the quality of the software developed, and the quality of the software is ensured by creating multiple test cases for each method with all possible combination of test cases. This also improves the productivity by generating the test cases automatically and any changes and new requirements need only a single click to run the test case creator. By reducing the unit test cycle time, developer can be productive and more innovative, auto generated test cases also improves the built in quality of the software’s developed. This tool can be used later for testing the UI and this tool can be integrated with the deployment and build system to automate the deployment process.

**VIII. REFERENCES**


