Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2014, 1(2): 69-74

Research Article

ISSN: 2394 - 658X

A Survey on Generation of Automated Test Data for Coupling Based Integration Testing

Amit A Kadam and Anant M Bagade

Department of Information Technology, Pune Institute of Computer Technology, Pune, India
kadamamit1811@gmail.com

ABSTRACT

In software engineering, software testing plays a vital role in improvement of software. In software testing, Test data generation is a standout amongst the most significant and crucial phases. Software testing is not possible without adequate test data. Software testing can be performed by using different test cases like, unit testing, integration testing, or system level testing. The first phase of testing is single module and we called as a Unit Testing, But by combining many unit testing module one other test type we can use i.e. Integration Testing. Integration testing tests the connections of different mechanism, when they are integrated together in precise application, for the smooth functionality of software arrangement. Many automated and manual test data generation techniques have been proposed for software testing. Coupling based testing is an integration testing which is based upon coupling relationships that exist among different variables within many other call sites are present. Up until now, test data generation approaches deal only with unit level testing. There is no work for software test information generation for coupling based testing. So, in this Paper, we proposed a novel methodology for automated test data generation based testing type classification and different algorithms that mostly used.

Key words: Software testing, test data generation, classification, automation, integration testing

INTRODUCTION

Software testing is a research going to give stakeholders with data about the quality of the result or service under test. Software testing can be perform by using different test cases like, unit testing, integration testing , or system level testing. The first phase of testing is single module and we called as a Unit Testing, Same thing for other test cases but not only the single module but also every modules like integration testing. Integration testing tests the connections of different mechanism, when they are integrated together in precise application, for the smooth functionality of software arrangement [1].

Integration level testing, tests the interaction of different components after integration with other components. System level testing treats the system as black box and checks the functionality of the framework as a whole. Integration testing is an important level of testing which verifies the different components interactions and message passing through interfaces. Unit level testing is a base for integration testing, if the units work correctly then different units are integrated together using different interfaces exposed by different components. Integration level testing verifies that the interfaces are correctly integrated and message passing through interfaces is correct. Integration testing is concerned with the interactions among components. Does a component call other components correctly? Are the right parameters with right types and ranges are passed? Does the called method return the proper type and the value is in the correct range? These questions are focus of the integration testing. Unfortunately, very little research has been done in area of coupling based integration testing test data generation using evolutionary approaches [1].

So, now automation has been achieved by various means of computers, usually in combination of others. Test data are an important part of test case, without test data test case execution is not possible. Research has explored several techniques for test data generation using evolutionary approaches [3-4]. A number of test data generation approaches have been developed and automated. Random test data generation, generates test data based on selective random inputs form some distribution. Path- oriented and structural approaches use the program's control flow graph for test data generation; they select a path, and use a technique such as symbolic execution for generation of test data. Goal-oriented test-data generation approaches select inputs to execute the selected goal, such as statement, condition coverage, decision coverage, irrespective of the path taken. Evolutionary test approaches use evolutionary algorithms i.e. genetic algorithm, for selection and generation of test data by applying evolutionary operators, i.e., crossover and mutation. Most of the work on test data generation has been done at unit level. Unit level test data generation involves the test data that executes the test case for unit level testing [3-4].

BASIC CONCEPTS

Concept Related to Software Testing

Software testing is the process of ensuring right software product to achieve full customer satisfaction. The following terms are mostly used for automated test data generation research.
Test data: Test Data are data which have been specifically identified for use in testing computer program.
Test case: Test case is a collection of conditions as well as variables under which a tester will conclude whether an application or software system is working properly or not.
Test oracle: The method for determining whether a software program or system has passed.
Test suite: A set of test cases is called test suite.
Test plan: is a document which contains all the information about the testing of all stages.
Test Automation: Developing software for testing a software product.
Path: sequence of nodes and edges. If we start from entry node and end at exit node then we call complete path [5].

Automated Test Data Generation

In General there are different reasons to automate test data generation task in software testing. Some of the most important reasons are as follows.

Reducing number of test cases: Generation efficient test cases are the essential identification for simplifying the test work and improving the test efficiency. The test work is inefficient because of the great number of the initial test cases, so some Automation algorithms are needed to optimize the test cases [3].

Reducing software testing cost: During testing phase the cost can increase more than the expected value due to inappropriate test data. These inappropriate test data cause wastage of organizational resources as well as time. There is a need to minimize the cost for getting an satisfactory product [6].

Reducing human errors: In order to find out how a test case is valid there is no definite mechanism. It basically depends on the testers understanding of the requirement. In this process there are lots of human errors and testers basic knowledge taken into concern. This leads to the addition of bugs in the system after testing. To overcome this problem, automated test data generation phase should be measured [7].

Increasing quality of software products: In general manual testing is becoming a bottleneck and is a frequent cause of project delays especially for large programs. Therefore, automated test data design has become important to ensure the quality of present day large software products [8].

There are some challenges in automating test data generation are as follows:
1. What types of techniques are there available at our clearance?
2. What are the characteristics different algorithms?
3. How to determine which technique is appropriate for what type of test data generation task?
4. Which methods can be used to make development in what aspect of the essential difficulties in test data generation?
5. When to apply test data generation in life cycle of software?
6. What should be the inputs for a test data generation?

CLASSIFICATION BASED ON TESTING TYPE

There are different types of software testing approaches are available. With respect to the fact that test data generation is an important phase in software testing, test data generation approaches directly depends on software testing types. There are different approaches available that represents a classification for different automated test data generation approach based on testing type as shown in figure 1.

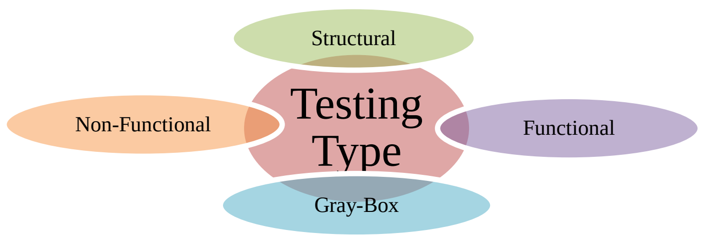

Fig. 1 Search based techniques

Structural Based Testing

In this approach test data are generated using system source code or control flow graph of the program. The main aim of structural testing is to cover a test adequacy criterion.

Advantages

- It is logical
- It is simply implementable

Disadvantages

- Generated test data is typically done very late in the software development life cycle.
- Errors exposed are very complex and costly to correct, since changes affect large portion of the design, implementation, and testing procedure [9].

Application

- Automated test data generation for functions in procedural programs such as triangle classifier function, bubble sort function etc.
- Automated test data generation for some significant classes in object oriented programs.

Functional Based Testing

In this approach test data are generated by using system specification. The goal of functional testing is to test the functionality of the software under test.

Advantages

- The testing can design much earlier in the software design process.
- Inconsistencies and ambiguities in specification are found earlier by the software tester.

Disadvantages

- Specifications are difficult and therefore difficult to understand.
- Implementation should exactly correspond with specification.
- There is need for formal specification of the system, which is tough to represent an exact identification of system specification in real application.

Application

- Automated test data generation for functions web or desktop applications such as banking system, employment system [10] etc.

Gray-Box Based Testing

In this approach both the structural and functional information are used for generation of test data.

Advantages

- It takes the benefits of both structural and functional based testing.

Disadvantages

- High complexities.

Application

- Automated test data generation for detecting run time errors called exceptional conditions in real time applications.

Non-Functional Based Testing

Non-functional testing is a trying of "how" the framework functions. Non-functional testing may be used at the all levels of testing. The term non-functional testing describes the tests required to measured characteristics of the systems and software that can be qualified on the verifying scalability of the system.

Advantages

- It takes in to the account other behaviors despite of logical behaviors.

Disadvantages

- It is very complicated because it depends both on software and hardware features Application
- Automated test data generation for execution testing of real time system[11]

CLASSIFICATION BASED ON ALGORITHM

The classification for automated test data generation approaches based on the variety of algorithm they used.

Random Based Testing

The test cases and events sequences are generated randomly. This algorithm is mostly used to analyze the efficiency of other algorithms in comparison with it.

Advantages

- It is easy to implement
- It require less cost
- It work well for simple program
- It is good at exercising systems when the source code and the specification of the program under test are not available or incomplete.
- Simply generating a huge number of test cases automatically.

Disadvantages

- It doesn't use available information for test case generation
- May give up a large number of event sequences that are not lagel and hence not executable, wasting valuable resources.
- The test designer has no control over choice of event sequences
- They may not have acceptable test coverage[11]

Application

- It mostly used for analyze the efficiency of other algorithm in comparison with this algorithm.

Search Based Testing

In these approach the problem of generating the test data are considered as an optimization problem, and try to find optimize solution including best test set for problem under test. Advantage of search based testing result shows efficiency of this approaches and disadvantage is that it requires large spaces. Application for search-based method is used in web/desktop applications. In search based automated test data generation a variety of search algorithm to use for the purpose of test data generation. The Figure 2 shows these search based methods [11].

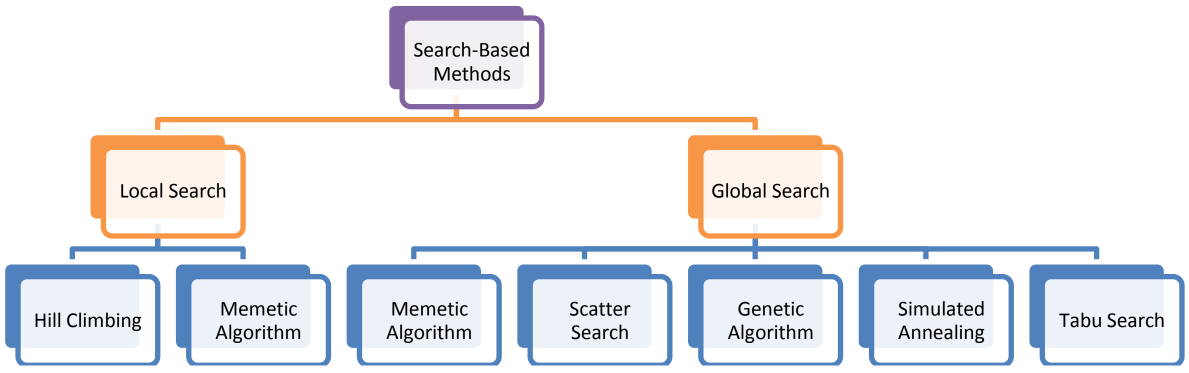

Fig. 2 Search based techniques

Hill Climbing

Hill climbing is a family of local search which is mathematical optimization technique. It is an iterative calculation that begins with an arbitrary answer for an issue, then endeavors to discover a superior arrangement by incrementally varying a single element of the result. In the event that the change delivers a superior arrangement, an incremental change is made to the new come about, rehashing until no further enhancements can be make. For instance, hill climbing can be applied to the TSP. It is not difficult to discover a beginning arrangement that visits all the cities however will be exceptionally poor contrasted with the ideal result. The calculation begins with such a come about and makes little upgrades to it, for example, exchanging the request in which two cities are visited. At the end, greatly shorter route is to be obtained.

Memetic Algorithm

The memetic algorithms (MAs) are meta-heuristics that use both global search and local search for instance GA with a hill climbing. It is motivated by the cultural growth. A meme is a unit of reproduction in educational transmission. The plan is to mimic the process of the growth of these memes. With the point of view of an optimization, a MA can be described as a population based meta-heuristic in which, every time an issue or offspring is generated, unless it reaches local optimum until a local search is applied to it.

Scatter Search

Scatter search is a search based evolutionary method that works on a set of solutions, called the reference set, which stores the best solutions that have been generated faraway. The solutions in this set are pooled in order to achieve new ones, so it trying to generate each time better solutions, according to excellence and diversity criteria [12].

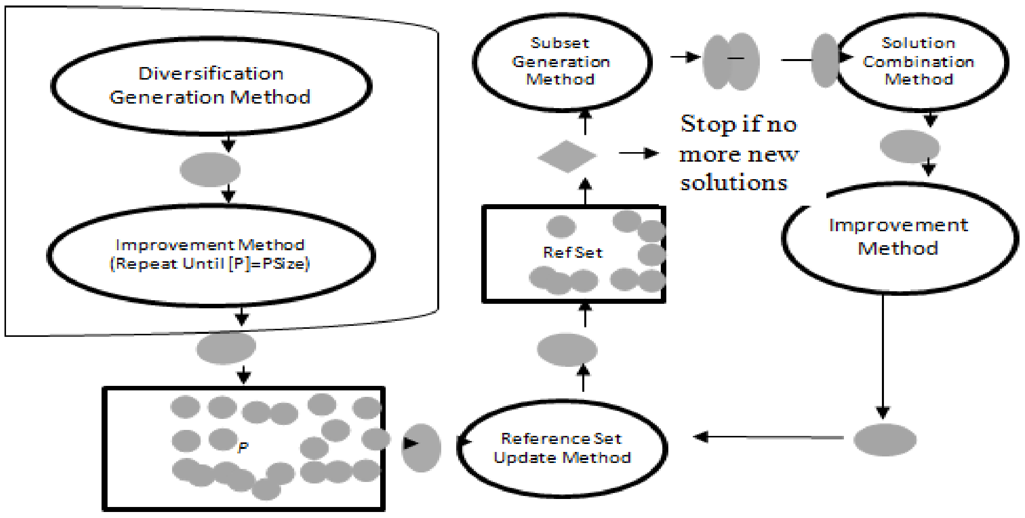
Diversification Generation Method
Improvement Method (Repeat Until [P]=PSize)
P
Subset Generation Method
Solution Combination Method
Stop if no more new solutions
Ref Set
Improvement Method
Reference Set Update Method

Fig. 3 Basic scheme of scatter search [12]

Tabu Search

Tabu search is a global search technique that solves combinatorial optimization problems by using meta-heuristic algorithm, such as the travelling salesman problem (TSP). Tabu search is technique that uses a local or neighborhood search method to iteratively move from a solution s to a solution s' in the region of s, until some stopping condition has been fulfilled. To search area of the search space that would be left unknown by the local search procedure, tabu search modifies the neighborhood structure of every result as the search progresses [6].

Genetic Algorithm

In the long-ago, evolutionary algorithms have been applied in many real life problems. Genetic algorithm is one such evolutionary algorithm. Genetic algorithm has emerged as a practical, robust optimization technique and search method. A Genetic algorithm is a search algorithm that is motivated by the way nature evolves type using natural selection of the fittest individuals. The possible solutions to problem can be solved which is represented by a population of chromosomes. A chromosome is a string of binary digits and each digit that makes up a chromosome is called a gene. This primary population can be totally random or can be created by hand using processes such as greedy algorithm. The pseudo code of a basic algorithm for GA is as follows [13] -

```
Initialize (population)
Evaluate (population)
While (stopping state not satisfied)
{
        Selection (population)
        Crossover (population)
        Mutate (population)
        Evaluate (population)
}
```

Simulated Annealing

It is one type of global search technique which is simulated annealing. It samples the whole area and improves the result by recombination in some form. In simulated annealing a value y1, is chosen for the result y, and the result which has the minimal cost or objective function, E, is chosen. Cost functions define the comparative and desirability of particular solutions. Minimizing the objective function is usually referred to as a cost function; whereas, maximizing is usually referred to as fitness function [11].

Table 1 Comparison between Automated Test Data Generation based on Testing Type

Approaches	Input	Used for real time system	Complexity
Structural Based Testing	Code or CFG	No	Low
Functional Based Testing	Specification	No	High
Gray-Box Based Testing	Code or CFG or Specification	Yes	High
Non-Functinal Based Testing	Software and hardware features	Yes	High

PROPOSED SYSTEM

The proposed system consists of two main parts. The first is a source code analyzer that identifies the coupling sequence/path from the .class file as input. The second part is the test data generation. Test data generation generate test data using genetic algorithm for coupling sequences/path generated in first part of our system for automatic test path generation using coupling information and integration testing criteria[1] and then based on the test data generated it provide the solution. The following Fig. 4 shows the proposed system.

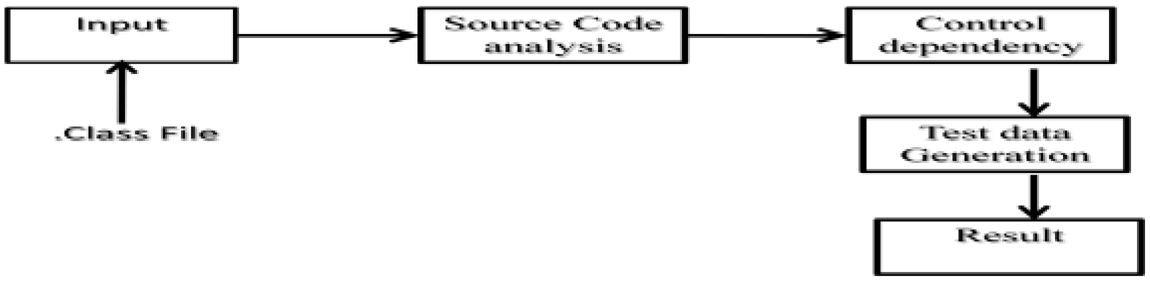

Fig. 4 Proposed system architecture

CONCLUSION

This paper provides an overview of the automated test data generation by using different testing type classification and based on algorithms which are an important area of research for reducing cost of software development. Test data generation is used to satisfy functional, non functional and business requirements. Some non functional requirement testing can be done only by automation; where manually it is not possible. The paper emphasizes the basic concepts of automated test data generation. It also focuses much on the test data generation classification techniques and methodology. We feel that these concepts are mandatory to perform research in the area of automated test data generation whether it is conventional programming or modern programming.

Acknowledgements
I would like to take this opportunity to thank all the people who supported me during the course of this work. Especially my mentor Associate Prof. A. M. Bagade, without his efforts this work wouldn’t have been completed.

REFERENCES

[1] Shaukat Ali Khan and Aamer Nadeem, Automated Test Data Generation for Coupling Based Integration Testing of Object Oriented Programs using Evolutionary Approaches, *IEEE 10th International Conference on Information Technology*, **2013**.
[2] Ankur Pachauri and Gursaran, Test data generation for Unit testing of Java programs Using JML and Genetic Algorithm, *International Journal of Software Engineering Research and Practice*, **2011**.
[3] Lee Copeland, *A Practitioner’s Guide to Software Test Design*, STQE Publishing, **2004**.
[4] Boris Beizer, *Software Testing Techniques*, International Thomson Computer Press, **1990**.
[5] Hitesh Tahbildar and Bichitra Kalita, Automated Software Test Data Generation: Direction of Research, *International Journal of Computer Science & engineering Survey*, **2011**, Vol 2.
[6] Y A Sharma, A Jadhav, P R Srivastava and R Goyal, Test Cost Optimization using Tabu Search, *J Soft Eng Appl*, **2010**, vol 3, pp 105 -156.
[7] V Rajappa, A Biradar and S Panda, Efficient Software Test Case Generation using Genetic Algorithm Based Graph Theory, *Proceedings of the First International Conference on Emerging Trends in Engineering and Technology*, **2008**, pp 298-303.
[8] S K Swain, D P Mohapatra and R Mall, Test Case Generation Based on State and Activity Models, *Journal of Object Technology*, **2010**, vol 9, pp 1- 27.
[9] W T Tsai, D Volovik, T F Keefe and M E Fayad, Automatic Test Case Generation from Relational Algebra Queries, *IEEE Transactions on Software Engineering*, **1990**, vol 16.
[10] A Alhroob, K Dahal, A Hossain, Automatic Test Cases Generation from Software Specifications, *e-Informatica Software Engineering Journal*, **2010**, vol 4.
[11] P McMinn, Search-Based Software Test Data Generation: A Survey, Software Testing, Verification & Reliability, **2004**, vol 14, pp 105–156.
[12] Raquel Blanco , Javier Tuya, Belarmino Adenso-Díaz , Automated Test Data Generation using a Scatter Search Approach, Information and Software Technology, **2009**, pp 708-720.
[13] Chayanika Sharma, Sangeeta Sabharwal, Ritu Sibal, A Survey on Software Testing Techniques using Genetic Algorithm, *International Journal of Computer Science Issues*, **2013**, Vol 10,Issue 1.