ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Junit Test Cases Generation by Generating Object Instances and Genetic Algorithm

Pranali Prakash Mahadik, Prof. Dr. D. M. Thakore

Research Scholar, Dept. of Computer Engg., Bharati Vidyapeeth Deemed University, Collage of Engineering, Pune, India.

Professor, Dept. of Computer Engg., Bharati Vidyapeeth Deemed University, College of Engineering, Pune, India.

ABSTRACT: Automating the process of test data generation is very essential to decrease software cost and development time.to generate test cases for object oriented code is very challenging because of the its features like abstraction, encapsulation, inheritance and visibility. Proposed system takes input as java classes and then it generate instances of classes and find out sequence of method calls and find out test target by using Genetic algorithm which is useful to find out optimum test target where GAs are also very useful and work efficiently when there is large search space and when domain knowledge is insufficient and finally generate Junit test cases.

KEYWORDS: Automatic test case generation; OOP challenges; Junit; Genetic algorithm.

I. Introduction

In software life cycle software testing is essential stage which includes test data generation which is most expensive and time consuming part [1].generating test data for object oriented code is challenging because of its features like visibility, abstraction, inheritance and encapsulation. Test cases generation is resources and effort consuming process for this type of code.

To overcome this drawback most of the approaches are developed like search based technique, random test data generation technique or symbolic execution. But from this all technique Search Based technique is most efficiently applied to solve the problem of test data generation[2], [3], [4].search based technique consider the state of classes.

Proposed system use search based technique along with generating instances of classes and call sequence of method call for this we are generating .csv files to store details of each class under test .then find out test target by using genetic algorithm because existing approaches randomly select the test target which may lead to less coverage of class and finally it generate test cases in Junit format.

II. Related work

To generate test data main 3 techniques are available,

A. Random Test Generation:

Random test data generation technique is also known as adaptive test data generation process. It provide better result in generation of test data. This technique randomly select input[4].this technique generate huge amount of test cases automatically in very fast manner. Drawback of this technique is that it cannot have sufficient test coverage and applicable for only simple code.

B. Symbolic Execution based techniques:

James King proposed symbolic execution is a technique in 1976 [5]. it include dynamic symbolic execution and concolic testing. Symbolic test data generation techniques [6, 7] assign symbolic values to the variables this technique executes a program with symbol. This technique can be used for many purposes, like as detection of bug, program verification, debugging of code, maintenance, and localization of fault [8].

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

C. Search-based techniques:

Search based technique name itself indicate some kind of coverage it has significant application in generation of test data because it has ability to produce software tests even in undesirable problems[9,10].this technique optimization technique to find best solution among all available solution. Main advantage of this technique is that test coverage is very efficient but it requires large search space.

III. Proposed algorithm

Proposed system developed to solve the problem of automatically generation of test cases.it generates Junit test cases which are very helpful for expert developers as well as for novice developer.

Proposed approach include following steps,

- It takes Input as java class file along with class path.
- Then generate instances by doing Pre-processing.
- Instantiate class under test and all other required classes.
- Perform sequence of method call.
- Apply genetic algorithm to reach test target.
- Generate test data.
- Display test cases in Junit format.Generate all the possible routes.

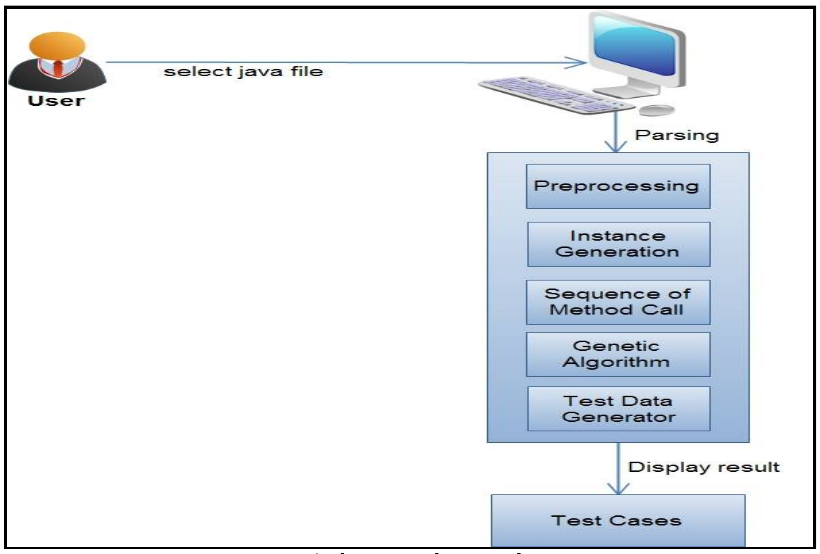


Figure 1. Architecture of proposed system

A. *Pre-processing stage:*

In this phase it parse the class under test i.e., java file and generate abstract syntax tree (AST) [11].this tree is modified according to method calls in class. Also in pre-processing stage it identify all branches wherein the particular method is called and identify branch modifiers.

B. *Generation of instances:*

To call method or constructor in object oriented code we need to generate instances of that particular class. This step is very important to achieve test coverage and also provide additional information for next step like sequence of method call.

In this step analysis of java class file is done I.e., it first finds out which type of class.

Class may be simple, atomic or container

- Atomic class – This type includes all string classes as well as basic or primitive class types.
- Container class- It includes List and Array, i.e., it is an object that can encompass other objects.

- Simple classes or other classes than container or atomic classes.

In this phase we generate instances of class by taking input as java class file and generate all needed elements then consider class type and based on it means-of-instantiation is selected.

C. *Generator of Sequences of Method Calls:*

This step required to put input java class in desirable state to achieve test target. It include following steps like,

- First generate instances for class
- Sequence of state modifier is generated
- Randomly select method from sequence of state modifier
- Generate target method

D. *Select test target by using genetic algorithm*:

GA has steps like selection, crossover, and mutation [12]. GA is used to generate test data because their robustness and suitability for solutions of different test tasks. [13], [14], [15].We are using genetic algorithm to find out the test target for the test cases.

Algorithm1. Genetic Algorithm to find out test target

```
Step 1: Let K be the number of elements.
Step 2:  Generate random numbers according to K.
Step 3:  Convert the random number generated in binary form.
Step 4:  Evaluate fitness for every member of the population.
Step 5: According to the evaluation, mutation and crossover are applied.
Step 6:  Convert the bit string of every fit individual into integer.
Step 7: Finally we get test target which is highest fitness value.
Step 8: End.
```

As an example, a simple genetic algorithm is given below:

```
{
Initialize population;
Evaluation of population;
While Criteria of Termination Not Satisfied
{
Select parents for reproduction;
Perform recombination and mutation;
 Population evaluation;
}
}
```

E. *Test Data Generator:*

Test-data generation is one of the most costly parts of the software Testing phase. It include following steps like,

- Analyse java file which is given as input to generate test data.
- To cover each branch select domain vector.
- Guide random generation to reach test target.
- Cover some uncovered branches.
- Translate set of test data into java file that contain test cases in Junit format.

E. *Generate Junit test cases:*

Junit is framework for unit testing which based on concept of first test then code. This frame work is very useful for all types of developer.it improves the productivity of programmer and reduce stress and time spent on debugging.

Features of Junit

- Junit is used for writing & running tests which is an open source framework.
- For testing expected results Provides Assertions.
- Test runners Provides for running tests.
- Junit tests allow writing code faster which improves quality
- Junit is simple, less complex and takes less time.

IV. RESULTS

Initially user need to select different java files to generate test cases figure 2. Describes how user can select java files along with class path. Proposed system has given option to select it and then click on Test File button so that system started to generate test cases.

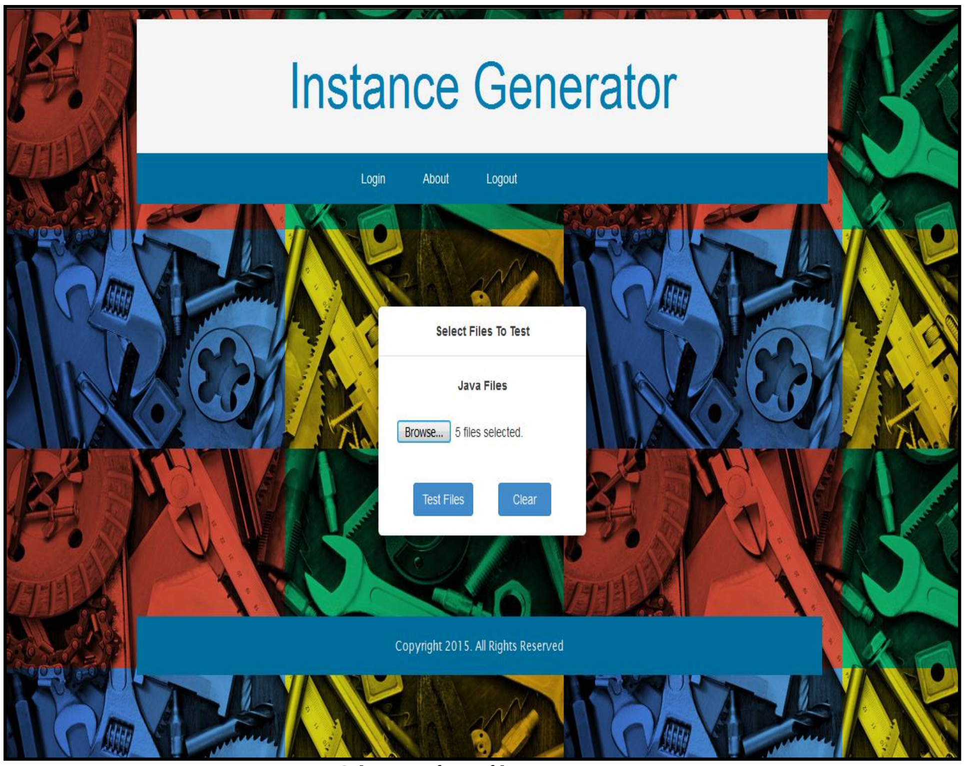


Figure 2. Selection of java files to generate test cases

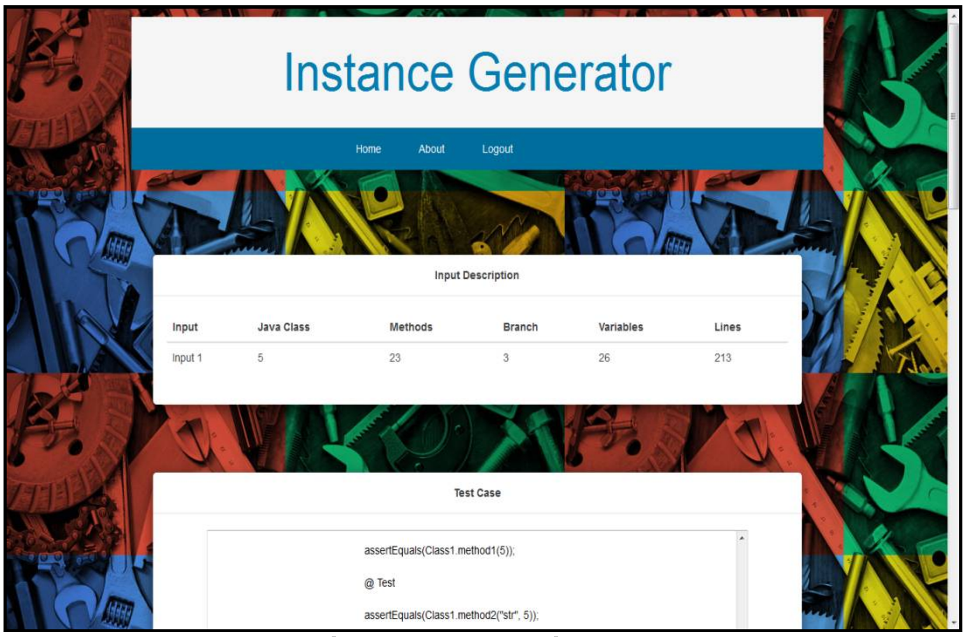
Figure3. Shows the input description i.e., the details of java file which we have provided as input in proposed system depending upon analysis it will produce output indicating the coverage like number of java class tested, number of methods, branches ,variables and lines along with test cases in Junit format.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016



Input Description

Input	Java Class	Methods	Branch	Variables	Lines
Input 1	5	23	3	26	213

Test Case

```
assertEquals(Class1.method1(5));
@ Test
assertEquals(Class1.method2("str", 5));
```

Figure 3.Code coverage input and Junit Test Cases

In figure 4.clearly shows the graphical representation of number of classes,methods,variables,branches and lines are get covered depending upon the mathematical analysis of proposed system.

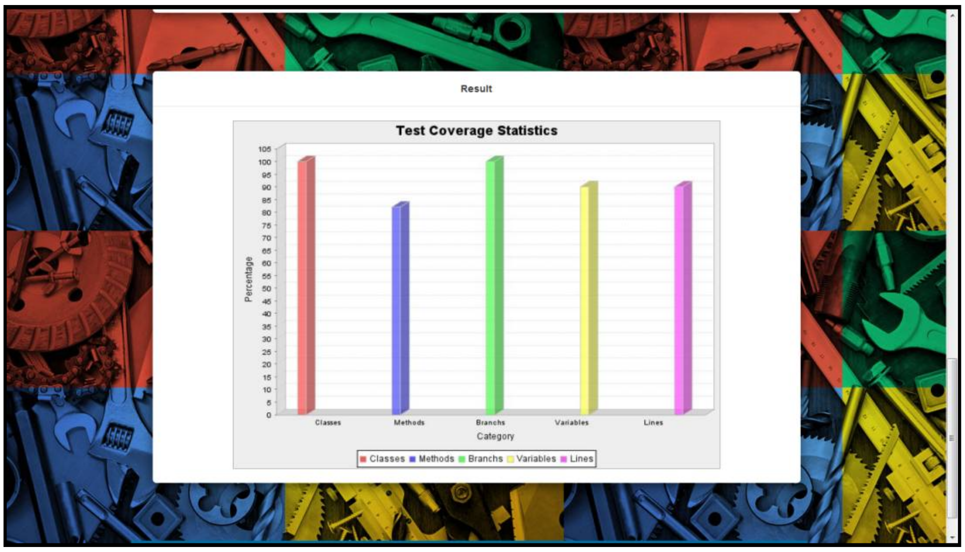


Figure 4. Result

V. CONCLUSION AND FUTURE WORK

Proposed system generates Junit test cases for java code and overcome the drawback of random searching. This approach is helpful for developers. use of genetic algorithm combined with existing approach improves the code coverage. In this system we are creating Junit test cases for the inputted java file which need to follow all the coding standards. We can enhance the capability to generate test case for unstandardized code. We can enhance this system to create the Junit for the Java framework like Struts, Spring etc.

REFERENCES

1. http://mit.bme.hu/~micskeiz/pages/code_based_test_generation.html (lastly accessed on April 05, 2016).
2. G. Fraser and A. Arcuri, “Evosuite: Automatic test suite generation for object-oriented software”, 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering, New York, NY, USA, (September 5-9, 2011), pp. 416–419.
3. P. McMinn, “Search-based software test data generation: A survey”, Journal of Software: Testing, Verification and Reliability, vol. 14, issue 2, (June 2004), pp. 105–156.
4. J. C. King, “Symbolic execution and program testing”, Communication ACM, vol. 19, no. 7, (July 1976), pp. 385–394.
5. Howden, William E.,”Symbolic testing and the DISSECT symbolic evaluation system‟, IEEE Transactions on Software Engineering, vol. 3, no. 4, (July 1977), pp. 266-278.
6. John Clarke, Mark Harman, Bryan Jones, ”The Application of Metaheuristic Search techniques to Problems in Software Engineering‟, IEEE Computer Society Press Vol. 42, No. 1, (2000), pp. 247-254.
7. L. A.Clarke and D. J. Richardson,” Applications of symbolic evaluation”, Journal of Systems and Software, (Feb 1985), pp. 15–35.
8. Tao Feng, KasturiBidarkar, ” Survey of Software Testing Methodology‟, vol. 25, no-3 , (2008) , pp. 216-226.
9. Voas ,Morell and Miller, "Predicting where faults can hide from testing‟, IEEE vol. 8 , pp. 41-48.
10. P.Tonella, “Evolutionary testing of classes”, SIGSOFT Software Engineering Notes, vol. 29, no. 4, (July 2004), pp. 119–128.
11. E. J. development tools (JDT), The JDT project provides the tool plug-ins that implement a Java IDE supporting the development of any Java application, including Eclipse plug-ins [Online],(2013), Available: http://www.eclipse.org/jdt
12. D. Berndt, J. Fisher, L. Johnson, J. Pinglikar, and A.Watkins, “Breeding Software Test Cases with GeneticAlgorithms”, IEEE Proceedings of the Hawaii International Conference on System Science, Hawaii. (Jan 2013), pp. 10.
13. B. T. de Abreu, E. Martins, F, de Sousa, “Automatic Test Data Generation for Path Testing Using a New Stochastic Algorithm”, XVIII SBES, (2005).
14. J. Wegener, K. Buhr, H. Pohlheim. “Automatic Test Data Generation for Structural Testing of Embedded Software System by Evolutionary Testing”, In GECCO, Vol. 2, (Jul 9 2002) ,pp. 1233-1240.
15. M.R .Girgis, “Automatic Test Data Generation for Data Flow Testing Using Genetic Algorithm”, Journal of Universal Computer Science, vol. 11, no.6, (2005), pp.898-915.