Automatic Test Case Generation in Object Oriented Programming

TARUN DHAR DIWAN 
ASSISTANT PROFESSOR 
DEPTT. OF ENGINEERING 
Dr. C.V.RAMAN UNIVERSITY 
BILASPUR (INDIA) 
taruncsit@gmail.com

GANESH SURYAVANSHI 
MPHIL SCHOLAR (COMPUTER SCIENCE) 
Dr. C.V.RAMAN UNIVERSITY 
BILASPUR (INDIA) 
ganesh.surya66@gmail.com

Abstract—This paper presents a new methodology to select test cases from regression test suites. The selection strategy is based on analyzing the dynamic behavior of the applications that written in any programming language. Methods based on dynamic analysis are more safe and efficient. We design a technique that combine the code based technique and model based technique, to allow comparing the object oriented of an application that written in any programming language. We have developed a prototype tool that detect changes and select test cases from test suite.

Keywords—Regression testing, Model based testing, dynamic behavior.

1. Introduction

Software maintenance is an expensive phase accounting for near 60% of overall cost of software life cycle expenditure [3]. Regression testing is an important step in software development to ensure that modifications do not break previously working functionality. However, regression testing is often expensive and time consuming. Regression test suites can be very large, e.g. including tens of thousands of test cases requiring days or weeks to execute [1]. Regression test selection is the activity that choosing from an existing test set, test cases that can and need to be rerun to ensure that changed parts behave as intended and the changes did not introduce unexpected faults. Reducing the number of regression test cases to execute is an obvious way of reducing the cost associated with regression testing. The main objective of selecting test cases that need to be rerun is to identify regression test cases that exercise modified parts of the system. This is referred to as safe regression testing as, it identifies all test cases in the original test set that can reveal one or more faults in the modified program [12]. There are many techniques that handle regression testing, some of them based on source code and other based on design. The techniques that based on source code are more safe and easy to make. But, it requires that the changes be already implemented. These techniques are very specific to the programming language used to develop the software. Where, if an application is built using functional languages such as C, hence, it is not suitable to analyze applications built using C# and Java because the tool cannot identify indirect changes due to object oriented features of these languages like dynamic binding, exceptions etc. Other techniques that based on specification are more general where the designs are represented using the Unified Modeling Language (UML) that independent on programming language. But, some changes to the source code may not be detectable from UML documents so cannot detect all test cases for the changes. In this paper we present a new approach that overcomes these shortcomings has been proposed. The approach is based on combining the code based technique and model based technique together to generate a safe and general regression test selection technique. Our approach capture and analyzing the dynamic behavior of the software applications from UML diagram. Then identify the impact of changes made to software, and based on this it selects test cases to be executed. These test cases are fewer in number when compared to the complete system test suite.

2. A SURVEY OF TESTING TECHNIQUES FOR OBJECT ORIENTED SYSTEMS:

Most research on object-oriented (OO) paradigms has been focused on analysis, design, and programming fundamentals. Testing the systems that are created with these paradigms has been considered an afterthought. Traditional testing techniques must be evaluated to determine if they are still useful with respect to object oriented systems, and new techniques must be developed.
3. Framework

Typically regression test selection techniques are either code-based or model-based. Code-based techniques use the information obtained from two different versions of the code to analyze the change impact and select the tests. In the case of model based techniques, change information is obtained through two versions of models constructed during the requirements analysis phase or system design phase.

4. RELATED WORK

Code based techniques [2], [5], [6], [7], [11], [12] select tests based on changes made to two versions of the code. These techniques are very specific to the programming language used to develop the code. Chianti [10] and JDiff [5] are comprehensive techniques for managing changes in Java programs. Chianti selects regression tests after analyzing the change impact analysis whereas JDiff performs only change impact analysis. As both these tools analyses the changes at statement level and are specific to Java programming language, hence, they are neither generic nor efficient. Model-based techniques [3], [4], [8], [9] are based on UML design models used during the design phase of the system. Reference [15] use UML activity diagrams to detect changes in design and then use a traceability matrix between activity diagram and the test suite. It covers activities at an abstract level and does not cover the attributes of a class. Also, it does not support object-oriented features. Reference [8] proposes a regression testing technique based on UML sequence and class diagrams. Their approach does not take into account the pre and post conditions of the operations which affect behavior of a class. Also, their approach does not handle Concurrency.

5. Challenges

Regarding color feature to image processing below challenges are targeted.
1. A main problem with testing object-oriented systems is that standard testing methodologies may not be useful.
2. The process of testing OO software is more difficult than the traditional approach, since programs are not executed in a sequential manner.

6. Significant

The aim of this paper is to study various established as well as emerging testing techniques, with special focus on those for object-oriented software.

7. Object

The objective of this paper is design and development of an automated testing tool for object-oriented software and develops a tool which is based upon the techniques which are most suitable due to their effective applicability to OO programs.

8. Used Method

6.1 Our Regression Test Selection Technique

Our proposed approach to regression test selection is based on changes made to software specification that represented in UML diagram and code that represented in any programming language. Our approach consists of three functions as shown in Fig. 1. These three functions are (1) Capture dynamic behavior, (2) Identify changes, (3) Select regression test suite. Each of these functions has been described in details. Fig. 1 Block diagram of our approach

A. Capturing Dynamic Behavior of the Application

Dynamic behavior of software is a set of interactions among system components along with their invoked classes/functions across all application processes. We captured dynamic behavior of the system from UML Class diagram and Sequence diagram. The captured behavior is modeled into Interclass Relation Graph (IRG) and Functional Interaction Graph (FIG). The Interclass Relation Graph (IRG) for a program is a triple \( \{N, IE, and UE\}\):

- \( N \) is the set of nodes, one for each class.
- \( IE \) is the set of inheritance edges. An inheritance Edge between a node for class \( C1 \) and a node for Class \( C2 \) indicates that \( C1 \) is a direct sub-class of \( C2 \).
- \( UE \) is the set of use edges. A use edge between a node for class \( C1 \) and a node for class \( C2 \) indicates that \( C1 \) contains an explicit reference to \( C2 \) Program P public.
6.2 Regression testing
Testing modified software to ensure that changes are corrects and do not adversely affect other parts of the software. Make use of existing test cases developed for previous versions of the software. May have to create new test cases as well.

6.3 Why is Regression Testing Important?
- Software is buggy
- Software is modified over time
  a. Adding new functionality
  b. Improving performance

6.4 Experiment of Object Oriented Based Regressing Testing Example

```plaintext
read(x, y);
if (y < 0) then
  power = -y;
else
  power = y;
endif
z = 1;
while (power != 0) do
  z = z - y;
power = power – 1;
endwhile
if (y < 0) then
  z = 1 / z;
endif
result = z;
write(result);
```

Program takes integer input (x, y) and should output \( xy \)

These 2 tests succeed and cover all branches:
- \( (x=2, y=-1) \)
- \( (x=1, y=0) \)

Now try a third test case:
- \( (x=3, y=3) \)

Before Program Modification:-
read(x, y);
if (y < 0) then
    power = -y;
else
    power = y;
endif
z = 1;
while (power != 0) do
    z = z - y;
power = power – 1;
endwhile
if (y < 0) then
    z = 1 / z;
endif
result = z;
write(result);

<table>
<thead>
<tr>
<th>Test Case (x, y)</th>
<th>Expected Output</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2, -1)</td>
<td>½</td>
<td>1/2</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(2, 0)</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>(3, 3)</td>
<td>27</td>
<td>-8</td>
</tr>
</tbody>
</table>

Test case (3, 3) fails!
Possible fix: when raising to a power, values should be multiplied, not subtracted!
Change line * to be:
z = z * y;

After Program Modification

read(x, y);
if (y < 0) then
    power = -y;
else
    power = y;
endif
z = 1;
while (power != 0) do
    * z = z * y;
power = power – 1;
endwhile
if (y < 0) then
    z = 1 / z;
endif
result = z;
write(result);
Automatic Test Case Generation in Object Oriented Programming

7. Analysis

Selecting Existing Test Cases:-
Which existing test cases should be used for regression testing?
Option 1: Re-run every existing test case
Assuming unlimited time and resources, this is best Problem: we don’t have unlimited time and resources!
Option 2: Re-run a subset of existing test cases but which existing test cases are likely to expose regressions in the software?

8. Output of Experimental
This presentation will describe some approaches to identify a subset of existing test cases to use for regression testing.

<table>
<thead>
<tr>
<th>Test Case (x, y)</th>
<th>Expected Output</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2, -1)</td>
<td>½</td>
<td>-1</td>
</tr>
<tr>
<td>(1, 0)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(3, 3)</td>
<td>27</td>
<td>27</td>
</tr>
</tbody>
</table>

Test case (3, 3) now succeeds!
Uh oh – test case (2, -1) fails
Regression testing should reveal this (correct fix: line * should be: z = z * x)

9. Result and Discussion
Focus of earlier regression testing research. Achieve required coverage of the modified program with minimal re-work. Focus of this research paper Verify that behavior of modified program is unchanged, except where required by the modifications. Identify test cases in the existing test suite on which the original and modified programs may produce different outputs. Modified program through two assumptions first is no statements are added to or deleted from the program and second no changes are made to the left hand side of assignment statements. These two assumptions will be relaxed later.

10. Conclusion
If a statement is not executed by a test case, it cannot affect the program output of that test case, not all statements in the program are executed by each test case.

11. Future work
Our future research would focus on investigation of techniques that automatically identify major changes made to code and generate test cases that validate these changes.

REFERENCES


AUTHOR BIOGRAPHIES

Tarun Dhar Diwan Received Master of Engineering (Computer technology and application) from Chhattisgarh swami Vivekananda technical university –Bhilai, India, and Master of Philosophy (Gold Medal list) from Dr.C.V.Raman University- .He is currently an Assistant Professor at the Dr.C.V.Raman University-bilaspur, India his current research work artificial intelligent, image processing, Software Engineering