ISSN:0975-9646

Object Oriented Software Testing

P.D.Ratna Raju, Suresh.Cheekaty, HarishBabu.Kalidasu

Priyadarshini Institute of Tech& Science
Tenali; Guntur(Dist),INDIA

***Abstract:*A test case is a set of conditions or variable and input that are developed for a perticular goal or objective to be achieved on a certain application to judge its capabilities or features.By an automated testing tool, we mean a tool that automates a part of the testing process. It can include one or more of the following processes: test strategy generation, test case generation, test data generation, reporting and results. By Object- oriented software can mean a software designed using OO approach and implemented using a OO language.As describe later, there is a lack of specifications as compared to program code is that specifications are generally correct where as code is flawed. Moreover, with software engineering principles firmly established in the industry, most of the software developed nowadays follows all the steps of Software Development Life Cycle (SDLC). For this work, UML specifications created in Rational Rose are taken. UML has become the de- facto standard for analysis and design of OO software.Testing of OO software is different from testing software using procedural language. Several new challenges are posed. In the part most of the testing methods for testing OO software were just a simple extension of existing methods for conventional software. However, they have been shown to be not very appropriate. Hence, new technology have bee this tool provides features for testing at unit (class) level as well as integrating level. Further a maintenance-level component has also been incorporated. Results of applying this tool to sample Rational Rose files have been incorporated, and have been found to be satisfactory.. Testing is conducted at 3 levels: Unit, integration and system. At the system level there is no difference between the testing techniques used for OO software and other software created using a procedural language, and hence, co It might take more than one test case to determine the true functionality of the application being tested. Every requirement or objective to be achieved needs atleast one test case.some software development methodologies like Rational Unified Process(RUP) recommend creating atleast two test cases for each requirement or objective ; one for performing testing through positive perspective and the other through negative perspective.**

Keywords: Testing, Unit, integration, system, UML, control flow graph, class, object, SDCL, Object-oriented, state transition diagram, design, analysis, implementation, black-box, white-box.

I. INTRODUCTION:

Test Case Structure

A formal written test case comprises of three parts-

A. **Information:** Information consists of general information about the test case. Information incorporates Identifier, test case creator, test case version, name of the test case,purpose or brief description and test case dependencies.

B. **Activity:** Activity consists of the actual test case activities.Activity contains information about test case environment,activities done at test case initialization,activities to be done after test case is performed,step by step actions to be done while testing and the input data that is to be supplied for testing.

C. **Results:** Results are outcomes of a performed test case.Result data consists of information about expected results and the actual result.

Designing Test Cases

Test cases should be designed and written by someone who understands the function or technology being tested. A test case should include the following information-

*Purpose of the test
*Software requirements and Hardware requirements (if any)
*Specific setup or configuration requirements
*Description on how to perform the task(s)
*Expected results or success criteria for the test

Designing test case can be time consuming in a testing schedule,but they are worth giving time because they can really avoid unnecessary retesting or debugging or at least lower it.Organization can take the test case approach in their own context and according to their own pespective.Some follow a general step way approach while others may opt for a more detailed and complex approach. It is very important for you to decide between the two extremes and judge on what would work the best for you.Designing proper test cases is very vital for your software testing plan as a lot of bugs,ambiguities,inconsistencies and slip ups can be recovered in time as also it helps in saving your time on continuous debugging and re-testing test cases.

Software testing is a phase of SDLC that entails much effort, time out and cost. Often, testing phase is the single largest contributor towards the whole development time. Testing can not only uncover bugs in the program, but also flaws in design of the software. To make the testing phase quicker, easier and more efficient, automated testing tools are being used. These tools help in test case generation, reporting results and variance from expected once(if any) , bugs in code and each other flaws. Usage of these tools speed up the testing process and also ensures reduction in the probability of a bug/error being uncovered later. However application of these automated testing tools in software testing has its own disadvantages, namely, learning the tool to use it, adapting it to your purpose, and also the tool may not provide specific functionality which you may desire.

Object-oriented testing essentially means testing software developed using object oriented methodology.

The objectives of developing the testing tools for software testers and maintainers are:

(1) To help them understand the structures of, and relation between, the components of an OO program.

(2) To give them a systematic method and guidance to perform OO testing and maintenance.
(3) To assist then to find better test strategies to reduce their efforts.
(4) To facilitate them to prepare test cases and test scenarios, and
(5) To generate test data and to aid them in setting up test harnesses to test specific components.
The target users for Testing Tools are mainly software testers and maintainers. As the tools would provide valuable insight into programmer's structure and behavior plus automate the testing process to a certain extent, it would be highly useful for testers. Also the tool would be beneficial to maintainers who would like to study change impact (here they will be aided by the program's analysis done by the tool), and perform regression testing.

II. OBJECTIVE

The objective of this paper is: design and development of an automated testing tool for object-oriented software. the aim of this paper is to study various established as well as emerging techniques, with special focus on those for object-oriented software; and develop a tool which is based upon the technique which are most suitable due to their effective applicability to OO program.

III. THE TEST MODEL AND ITS CAPABILITES**:**

The tools for automated testing are based upon certain models of software/programs and algorithms. This mathematically defined test model consists of following types of diagrams:
1. The class diagram (object relation diagram)
2. The control flow graph (of a method), and
3. The state transition diagram (of a class)

A. *CLASS DIAGRAM***:**

A class diagram or an object relation diagram (ORD) represents the relationships between the various classes and its type. Types of relationships are mainly: inheritance, aggregation and association.

B. *CONTROL FLOW GRAPH*:

A control flow graph represents the control structure of a member function and its interface to other member functions so that a tester will know which ata is used and/or updated and which other functions are invoked by the member function.

C. *STATE TRANSITION DIAGRAM*:

which are shared among A STD or an object state Diagram (OSD) represents the state behavior of an class. Now the state of a class is embodied in its member variables its methods. The OSD shows the various states of a class (Various member variable values), and transitions between them(method invocations).

IV. BASED ON SOFTWARE DESIGN/SPECIFICATION:

These diagrams are taken from the design models prepared as part of software Development process. UML (Unified Modeling Language) has become the defacto standard for object-oriented analysis and design (OOAD). UML provides features for specifying all the above types of diagrams. Rational Rose Suite is the most widely used
Methodology adopted:
<u>For carrying out this paper, following methodology has been adopted:</u>
1 .Literature survey: This involves study of existing techniques and strategies, with special emphasis on object-oriented testing.
2 .Analysis of problem: this incorporates analyzing the problem. Out of literature survey emerged; the right techniques and tactics for object oriented software testing also existing methods have been modified upon where ever necessary.
3 .Software tool development: since the ultimate objective of this paper is to develop an automated testing tool, all the software development has been followed.
(1)Analysis
(2)Design
(3)Implementation
(4)Testing
(5)Iterative process

Existing Testing Techniques Surveyed:
Black Box Testing:
(1)Random testing
(2)Equivalence partitioning
(3)Boundary value analysis
(4)State transition-based testing
White Box Testing:
(1)Basic path testing
(2)Loop testing
(3)Mutation testing
(4)Data flow-based testing

Testing Techniques To Testing Object-Oriented Software:
Certain subset of testing techniques covered in the study can be favorably applied to object-oriented programs. At various levels of testing of object oriented software, techniques which can be applied are:
1.unit testing
2.class testing
3.method testing
4.integration testing
5.system testing

Challenges To Testing Object-Oriented Systems:
A main problem with testing object- oriented system is that standard testing methodologies may not be useful. Smith and Robson say that current IEEE testing definition and guidelines cannot be applied blindly to OO testing, because they follow the Von Neumann model of processing. This model describes a passive store with active processor acting

upon store. It requires that there be an oracle to determine whether or not the program has functioned as required, with comparison of performance against a defined specification. "They also present the following definition of the testing process: "the process of existing the routines provide by an object with the goal of uncovering errors in the implementation of the routines or the state of the object or both."
Smith and Robson sat that the process of testing OO software is more difficult than the traditional approach, since programs are not executed in a sequential manner. OO components can be combined in an arbitrary order; thus defining test cases become a search for the order of routine that will cause an error. Siepmann and Newton agree that the state-based nature OO systems can have a negative that will cause an error. Siepmann and Newton state that the iterative nature of developing OO systems requires regression testing between iterations. Smith and Rabson state that inheritance is problematic, since the only way to test a Subclass is to flatten it by collapsing the inheritance structure until it appears to be a single class. When this is done, the testing effort for the super class is not utilized; therefore, duplicate testing takes place

V. 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.

VI. COMPONENTS OF THE OO TESTING TOOL:

The tool for automated testing of OO programs has the following components/features:
1. GUI
2. Import File Feature
3. Change Impact Identifier for classes
4. Maintenance Tools
5. Logging results
6. Diagram Displayer
7. Class Diagram
8. State Transition Diagram
9. Control Flow Graph
10. Test Tools:
    (i) Test order generator for testing of classes at integration level
    (ii) Test case generator for testing classes
11. Basis path generator for member functions/methods

VII. LATEST RESEARCH:

The latest research in the field of object- oriented software testing. Tonella [20] proposes a method for evolutionary testing of classes. In this paper, a genetic algorithm is exploited to automatically produce test cases for the unit testing of classes in a generic usage scenario. As, object oriented programming promotes reuse of classes in multiple contexts, the unit texting of classes cannot make too strict assumptions on the actual method invocation sequences, since these vary from application to application . Traore [21] discusses a test model for object-oriented programs, based on formal specifications like UML, built from user requirements. Pezze & young [22] have highlighted some important issues to be considered while testing object-oriented programs. Object oriented software requires reconsidering and adapting approaches to software test and analysis.

CONCLUSION:

*OBJECT ORIENTED TESTING LEVELS-UNIT&SYSTEM SAME AS TRADITIONAL LEVELS
*OBJECT ORIENTED INTEGRATION TESTING IS DIFFERENT AND MORE COMPLEX
*OBJECT ORIENTED OPTIMAL TEST ORDER SAVES
*TOOLS REQUIRED TO SCALE UP 00 TESTING
*LIMIT DESIGNERS TO STRAIGHT INHERITANCE (NO REDEFINING)

This paper dealt with design and development of an Automated Testing Tool for OO software. The tool mainly focuses on testing design specifications for OO software specification as compared to program code is that specification are generally correct

Development Life Cycle (SDLC) are adhered to. For this work, UML specifications are considered .UML has become the defector standard standard for analysis and design of OO software. UML designs created in Rational Rose are used by the tool as input. The main components of this tool are:

1. Test Order Generator for classes
2. Test Case Generator for State-based classes testing
3. Change Impact Identification for classes

FUTURE WORK:

Future work would include extending the tool to incorporate more functionality. Both testing and maintenance components can be added. Some additions can be:
1. Incorporating a fully functional Method Basis path generator module.
2. Providing both test case Generation as well as Execution. The user would be able to provide test data; and the test cases generated would be executed using the test data as input.
3. Reporting code coverage achieved after test set has been executed. Various test adequacy criteria like statement coverage, branch coverage, and path coverage can be reported upon.
4. Metrics: certain program metrics like lines of code (LOC), functions points, interfaces , etc.., can be reported upon.

REFERENCES:

[1] GAO, J.Z.Kung, D.Hsia, P.Toyoshmia, Y.Chen, C.”Object state testing for object-oriented programs” Computer software and Applications Conference, 1995 COMPSAC 95. Proceedings, Nineteenth Annual International, 9-11-Aug. 1995 pages: 232-238.

[2] Ugo Buy, Alessandro Orso ,Mauro pezze “ Automated Testing of Classes” August 2000 ACM SIGSOFT software Engineering Notes, Proceedings of the 2000 ACM SOFTWARE international symposium on software testing and analysis, Volume 25 Issue 5.

[3] Mary Jean Harrold , Gregg Rothermel “Performing data flow testing on classes ’’ December 1994 ACM SIGSOFT software Engineering Notes, PROCEEDINGS of the 2nd ACM SIGSOFT symposium on foundations of software engineering, volume 19 Issue 5

[4] Frank l, Elaine Weyuker “An applicable family of data flow testing criteria ’’ IEEE Transactions on software, vol. 14, no.10,1988.

[5]M. Smith and D.Robson “A Framework for testing object-oriented programs’’Journal of object-oriented programming, June 1994,pp.45-53.

[6] Doong, frank l “Case Studies on testing OO programs’’ Communications of the ACM, Vol.25, No.5, 1991.

[7] Doong ,frank l “ASTOOT approach to testing OO programs” ACM transactions on software Engineering and Methodology,vol.3,No.2,1994 .

[8]Kung, GAO, Hsia “Developing an OO testing and maintenance environment” Communications of the ACM, vol.38, No.10, 1995.

[9]Jorgensen, Erikson “object-oriented Integration Testing” Communication of the ACM, Vol.37, No.9, 1994.