Journal of Advances and Scholarly Researches in Allied Education
Vol. 13, Issue No. 2, July-2017, ISSN 2230-7540

Integration Object-Oriented Testing with UML

Fauja Singh*

Ravi Chowk, Purani Abadi, Sri Ganganagar, Rajasthan, India

Abstract – Today's world is Object Oriented. In this Object Oriented world, we have Object-Oriented Languages, Object-Oriented Analysis, Deigning, and Testing of Softwares. From History, the foremost popular Object-Oriented modeling approaches are Rumbaugh's Object Modeling Technique (OMT), which was better for Object-Oriented Analysis (OOA), and Grady Booch's Booch method which was better for Object-Oriented Design (OOD). In 1994 OMT and Booch Method were unified into Unified Modeling Language (UML). Later on, in 1996, Object-Oriented Software Engineering (OOSE) also unified with UML.UML has synthesized the notation of Booch Method, the Object Modeling Technique, and Object-Oriented Software Engineering by fusing them into single and widely usable modeling language. Object-Oriented Testing remains separate. Therefore the thrust behind this paper is to integrate Object-Oriented Testing with UML.

Keywords – Object-Oriented Testing, OOAD, OOSE, UML, etc.

----------X----------

1. INTRODUCTION

Object-oriented software is being developed in repeated additions. It tests the firms and interfaces of the components that are put together. Object-oriented Testing is often done on four different levels:

I. Testing one method.

II. Testing an object.

III. Testing sets of object-oriented components.

IV. Testing a whole system.

There are usually four levels of Testing for object-orientated systems counting on users approach, consisting of:

I. Method Testing.

II. Class Testing.

III. Interclass Testing.

IV. System Testing.

Smallest testable unit is the encapsulated Class. Traditional Testing focuses on input-process-output, whereas class testing focuses on each method. OO does not have an ordered control structure; that is why traditional top-down and bottom-up integration tests have little meaning.

OO Testing is analogous to the Testing of a standard system but is different. Once code has been created, OO testing begins with class testing. A series of tests are designed for those exercise class operations and examine whether errors exist together with class procedures and analyze whether flaws exist together. Class cooperates with other classes.

2. ISSUES OF OBJECT ORIENTED SOFTWARES

Testing within the case of conceptual software and therefore the issue of object-oriented software are-

2.1 Unit Testing within the OO context

Class Testing of OO software is equivalent to unit testing for conventional software.

2.2 Integration Testing within the OO Context

There are two different strategies for integration testing. The first, Thread Based testing, combines the classes needed to reply to at least one input or event for the system. Each thread is integrated and tested individually. The second integration approach, Use Based Testing, begins designing the system by testing those separate classes that use only a few server classes. After the independent classes are tested, the following classes, called Dependent Classes, are tested. Cluster Testing is one stride within the integration testing of OO software. Here, a cluster of collaborating lasses is exercised by designing test cases that plan to uncover errors within the collaboration.

2.3 Validation Testing within the OO context

At the validation or system level, the small print of the category connection disappears. The acceptance of OO software focuses on user-visible action and user-recognizable output from the system. to help in deriving validation tests, the tester should draw upon the use-cases that are a part of the analysis model. Traditional black-box testing methods are often wont to drive validation tests. Besides, test cases could also be derived from the object-behavior model, and therefore the event flow chart was created as a part of OOA.

Testing affects all stages of the software engineering cycle. At present, UML includes OOA, OOD, and OOSE. The thrust of this paper is to integrate OOT (Object-Oriented Testing) with UML.

3. INTRODUCTION TO UML

3.1 OOAD

Object-oriented analysis and design (OOAD) may be a software engineering method that models a system as a gaggle of related objects. Various models are often constructed to point out the static structure, dynamic behavior, and run-time deployment of those collaborating objects. Several different notations represent these methods, like the Unified Modeling Language (UML). Object-oriented analysis (OOA) applies object-modeling techniques to evaluate the functional requirements for a system. Object-oriented design (OOD) explains the analysis models to supply implementation specifications. OOA concentrates on what the system does, OOD on how the system does it.

3.1.1 Object-oriented systems

Objects make an object-oriented system. The performance of the system comes from the mixing of these objects. Integration of objects involves those sending messages to every other. Sending a message not the same as calling a function therein; when a destination object receives a message, it determines what function to bring the best service that message. An, and many various functions could also achieve an equivalent message one selected counting on the state of the target object.

3.1.2 Object-oriented analysis

Object-oriented analysis (OOA) looks at the matter domain, meaning to produce a theoretical model of the knowledge that exists within the field being analyzed. The results of the object-oriented analysis may be a description of what the system is functionally required to try to do during a theoretical model. It is also going to include some interface mock-up. Object-oriented analysis is to develop a model that describes computer software because it works to satisfy a group of customer-defined requirements.

3.1.3 Object-oriented design

Object-oriented design (OOD) transforms the theoretical model produced in object-oriented analysis to think about the constraints enforced by the chosen architecture and any non-functional technological. The result is a model of the answer domain, an in-depth description of how the system is made.

3.1.4 Object-Oriented modeling

Object-Oriented modeling, or OOM, maybe a modeling paradigm mainly utilized in programming. Before the increase of OOM, the effective prototype was procedural programming, which highlighted the utilization of prudent reusable code blocks that would stand on their own, take variables, perform a function on them, and return values.

The Object-Oriented paradigm assists the programmer in addressing the complication of a drag domain by taking the matter not as a gaggle of functions that will be executed but primarily as a group of related, interacting Objects. The modeling task then specifies, for a selected context, those Objects (or the category the Objects belong to), their corresponding set of fields and Methods, shared by all Objects of the category.

3.2 Object-oriented software engineering

Object-oriented software engineering (OOSE) is an object modeling language and technique. In 1992 OOSE was developed by Ivar Jacobson. It is the primary object-oriented design technique to use cases to make software design. Together with the Unified Modeling Language (UML), primary sources, concepts, and documentation from OOSE are fused in UML.

3.3 UML

Unified Modeling Language (UML) could also be a regulated general-purpose modeling language within the world of software engineering. The Unified Modelling Language (UML) is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software-intensive system under development. UML offers a typical because to imagine a system's architectural blueprints.

UML is used with all processes throughout the software development life cycle and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modelling technique (OMT), and Object-oriented software engineering (OOSE) by fusing them into one familiar and widely usable modeling language. UML aims to be a specific modeling language that can model concurrent and distributed systems. UML could also be a de facto industry standard evolving under the thing Management Group (OMG). OMG

initially involved information on object-oriented methodologies which can create a rigorous software modeling language. Many industry leaders have responded in earnest to help make the UML standard.

The Unified Modeling Language (UML) assists software analysts and designers visualize, document, and construct object-oriented systems. Three leading advocates of object-oriented methodology, Grady Booch, James Rumbaugh, and Ivar Jacobson, developed UML.

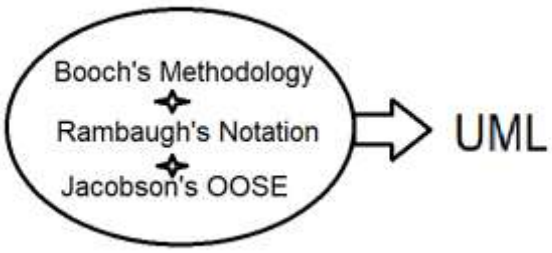

Figure 1

Following are the varied modeling diagrams UML notation provides:

- Use Case diagram
- Class diagram
- Object diagram
- Interaction diagrams – Sequence diagram, Collaboration diagram
- Activity diagram
- Statechart diagram
- Component diagram
- Deployment diagram

4. OBJECT-ORIENTED TESTING

The essential advantage of the object-oriented paradigm is that it provides a consistent structure for all components. This appears to upset the testing process; it will be exploited to support an object-oriented approach to Testing. To check a category, the programmer must be ready to undertake the subsequent activities:

- Create an instance of the category, i.e., an object, passing the acceptable parameters to the constructor
- Call the methods of the thing passing parameters and receiving results (c) examine the interior data of the item.

This can be practiced by creating a test program for every Class and, therefore, debug statements. However, it could even be achieved by the inclusion of appropriate mechanisms within the program development environment itself. This can eliminate the necessity for both test programs and, therefore, modify the category being tested.

An Object-Oriented system consists of several objects which communicate with one another. These objects contain both data and behavior, making them larger units than the individual routines that one works within a standard system development method. Object-oriented testing approach prevails among automated testing tools today thanks to:

- Vertical capability focus allowing to take advantage of specific features of the programming platform,
- Robustness of test scripts because of the power to spot properties and invoke actions of individual GUI components,
- Reasonable test result verification means,
- Easy to know for engineers conversant in the actual programming platform.

5. AN OBJECT-ORIENTED TESTING FRAMEWORK

The testing framework is meant to support the Testing of object-oriented class hierarchies.

5.1 The object test

Object testing is meant to check one Class. In this test, an Object is made, called, and queried to verify correct results. This sort of test is further sub-divided into:

Method tests: In each, one method is named to verify the proper operation.

Behavior tests: A series of methods are called to verify the proper behavior of a series of state transitions.

5.2 The collaboration test

The object-level test is merely the primary step in Testing for a reliable and robust class hierarchy. The subsequent step is to check objects together. The technique recommended here is to make a separate test object for every identifiable set of collaborating objects. A group of objects that implements a design

pattern, for instance, is a perfect (and obvious) candidate for a collaboration test suite. Differentiating an appropriate set of objects is by functionality: the set of objects that must perform a selected function must be tested to ascertain if they complete it. (Strictly speaking, these are two different sorts of Testing, but we lump them in together.)

6. AN OO TESTING APPROACH

The main advantage of working closely with other IT professionals is that they learn new skills from them, and therefore the best object developers will learn and adapt fundamental concepts from other disciplines. An example is class normalization, the object-oriented version of knowledge normalization, a set of straightforward rules for reducing coupling and increasing cohesion with object designs.

Testing techniques

Black-box Testing

Testing that validates the item being tested when given the acceptable input provides the expected results.

Class testing

The act of ensuring that a category and its instances (objects) operate as defined.

Class-integration testing

The act of assuring that the classes, and their instances, from some software, perform as defined.

Code review

A sort of technical review during which the deliverable being reviewed is an ASCII text file.

Component testing

The act of justifying that a component works as defined.

Coverage testing

The act of ensuring that each line of code is exercised a minimum of once.

Design review

A technical review during which a design model is tested.

Inheritance-regression testing

The act of working the test cases of the superclasses, both direct and indirect, on a liable subclass.

Integration testing

Testing to verify that several modules of software work together.

Method testing

Testing to verify away (member function) performs as defined.

Model review

An inspection, ranging from a proper technical review to an off-the-cuff walkthrough, by others who have not directly been involved in the model's event.

Path testing

The act of ensuring that each one logic path within the code is exercised a minimum of once.

Prototype review

A process by which users run through a set of use cases, employing a prototype as if it had been the natural system. The most goal is to check whether the planning of the prototype meets their needs.

Prove it with code

the most straightforward thanks to determining if a model reflects what is needed or what should be built is to create software that supported that model that shows that the model works.

Regression testing

Ensuring that previously tested behaviors still work, needless to say, after changes are made to the software.

Stress testing

The act of assuring that the system performs under high volumes of transactions, users, and load.

Technical review

a top-quality assurance technique during which the planning of an application is examined critically by a gaggle of peers. This process is usually mentioned as a walkthrough, an inspection, or a referee.

Interface testing

The interface (UI) testing ensures that it follows accepted UI standards and meets its wants. They are often mentioned as graphical interface (GUI) Testing.

White-box Testing

Testing to validate that particular lines of code work as expected. Also mentioned as clear-box Testing.

7. STRATEGIES UTILIZED IN OOT

To cover the strategies and tools related to object-oriented Testing

- Analysis and style Testing
- Class Tests
- Integration Tests
- Validation Tests
- System Tests

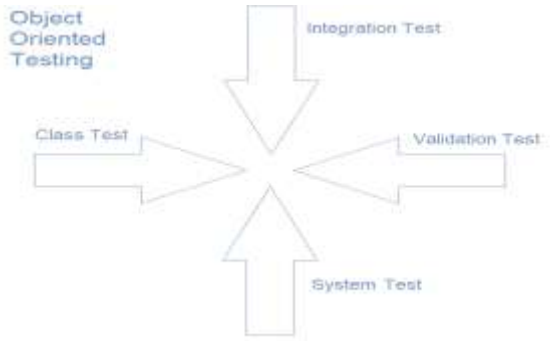

Figure 2

7.1 Analysis and Design:

Testing begins by validating the OOA and OOD models

- How can OOA models we tested (requirements and Use cases)?
- How can OOD models we tested (class and sequence diagrams)?

7.2 Class (Unit) Testing

Smallest testable unit is that the encapsulated Class. Test each operation as a part of a category hierarchy because its class hierarchy defines its context of Use.

Approach:

- Test each method (and constructor) within a category
- Test the state behavior (attributes) of the category between methods

How is class testing different from conventional Testing?

Traditional Testing centers on input-process-output, whereas class testing centers on each method, then composing sequences of methods to practice elements of a class. However, white-box Testing can still be applied.

Each test suit should contain:

- A list of messages and actions which will be applied as an outcome of the test
- A list of exceptions that will occur because the Object is tested
- A list of external conditions for setup (i.e., changes within the environment external to the software that has got to exist to conduct the test properly)
- Supplementary information will aid in understanding or implementing the test
- Automated unit testing tools facilitate these requirements.

White box tests:

- Basis track, state, data flow, and loop tests can all apply to individual practices but do not test intercommunications between methods.
- Identify methods applicable to a category.
- Define restrictions on their Use – e.g., the category should be initialized first.
- Identify a minimum test sequence – an execution sequence that defines the minimum life history of the category.

7.3 Integration applied three different incremental strategies

Thread-based Testing: integrates classes required to reply to at least one input.

Use-based Testing: combines classes asked by one use case

Cluster testing: integrates classes required to demonstrate one collaboration multiple Class Random Testing.

7.4 Validation

Are we building the right product?

Validation succeeds when software functions in a way that the customer will reasonably expect. Specialize in user-visible actions and user-recognizable outputs.

Details of sophistication connections disappear at this level Validation uses:

- Use-case scenarios from the software requirements spec
- Black-box Testing to make a deficiency list
- Acceptance tests through alpha testing at designer's site and beta testing at user's site with actual customers

8. CONCLUSION

Over the past decade, Grady Booch, James Rumbaugh, and Ivar Jacobson have collaborated to mix the most specific features of their OOA and OOD methods into a unified method. The result called the Unified Modeling Language [UML] had become widely used throughout the industry. In UML, a system is represented using five different "views" that illustrate the system from precisely different panoramas. The views present in UML are:

1. User Model View; This view represents the system [or product] from the user's perspective.
2. Structural Model View: Data and operations are inspected from within the system.
3. Behavioral Model View: This part of the analysis model represents the dynamic or behavioral aspects of the system. It also portrays the communications or collaborations between several structural components specified within the user model and structural model aspects.
4. Implementation Model View: The fundamental and behavioral features of the system are depicted as they are to be formed.
5. Environment Model View: The structural and behavioral aspects of the environment during which the system is implemented are represented.

In general, UML analysis modeling focuses on the user model and structural model, and structural model views of the system. UML design modeling discusses the behavioral pattern, implementation model, and environmental model aspects.

We do not have an Object-Oriented Testing Model in UML. That is why users adopt different testing models. This solution will help to form a typical testing model. The user would require that they ought to be using UML.

This study will help to mix Object-Oriented Testing with Object-Oriented Analysis, Object-Oriented Designing, and Object-Oriented Software Engineering into UML.

REFERENCES

[1] Bhadauria, Sarita Singh, Abhay Kothari, and Lalji Prasad (2011). “A Full Featured Component (Object-Oriented) Based Architecture Testing Tool" International Journal Of Computer Science Issues (IJCSI) 8.4: pp. 618-627.

[2] Gu, Dechang, Yin Zhong, and Sarwar Ali (1994). On Testing of Classes in Object-Oriented Programs‖ Proceedings of the 1994 Conference of the Centre for Advanced Studies on Collaborative Research

[3] Jain, Ajeet K. (2008). "Testing Polymorphism in Object-Oriented Programming." ICFAI Journal of Computer Sciences 2.4: pages 43-53. Johnson, Jr., Morris S. A Survey of Testing Techniques for Object-Oriented Systems, Proceedings of the 1996 Conference of the Centre for Advanced Studies on Collaborative research (CASCON '96)

[4] Khatri, Mrs. Sujata, Chhillar Dr. R. S. and Sangwan Mrs. Arti (2011). "Analysis Of Factors Affecting Testing In Object-oriented Systems," International Journal On Computer Science And Engineering 3: pp. 1191.

[5] Labiche, Y., Thevenod-Fosse, P., Waeselynck, H., and Durand, M.-H, "Testing Levels for Object-Oriented Software," Proceedings of the 22nd International Conference on Software Engineering, pages 136-145

[6] Turner, C.D., and Robson, D.J. (1993). "The State-Based Testing of Object-Oriented Programs," Software Maintenance, 1993 CSM-93, Proceedings., Conference on Software Maintenance pages 302-310, 27-30 Sep1993

[7] Mahfuzul Huda, Dr.Y.D.S.Arya, and Dr.M. H. Khan. "Measuring Testability of Object Oriented Design: A Systematic Review." International Journal of Scientific Engineering and Technology, Vol. 3, Issue 10, pp: 1313-1319 Oct 2014

[8] Briand, L. C., Labiche, Y., & He, S. (2009). Automating regression test selection based on UML designs. Information and Software Technology, 51(1), pp. 16–30. doi:10.1016/j.infsof.2008.09.010.

[9] Zheng, W., & Bundell, G. (2008). Contract-Based Software Component Testing with UML Models. Computer Science and its Applications, 2008. CSA '08. International Symposium on, 978-0-7695(13 - 15 October 2008), 83–102.

[10] Swati Tahiliani, Pallavi Pandit "A Survey of UML-Based approaches to Testing," International Journal of Computational Engineering Research, Vol.2 Issue.5, pp. 1396--1400.

[11] Nabil Mohamamed Ali Munssar and Dr. A. Goardhan (2012). "Comparison study between The traditional and The object-oriented approaches to Develop all project in software Engineering," International Journal Of Computer Science And Information Technology, Vol 3(1), pp. 3022--3028.

[12] David C.Kung and Pei Hsia "Object-Oriented Software testing-some Research and Development," computer Science and Engineering.

[13] Genero M., J. Olivas, M. Piattini and F. Romero (2001). "A Controlled Experiment for Corroborating the Usefulness of Class Diagram Metrics at the early phases of Object Oriented Developments," Proceedings of ADIS 2001, Workshop on decision support in Software Engineering, 2001.

[14] Kout, A., Toure, F., & Badri, M. (2011). An empirical analysis of a testability model for Object oriented programs. ACM SIGSOFT Software Engineering Notes, 36(4), 1. doi:10.1145/1988997.1989020.

Corresponding Author

Fauja Singh*

Ravi Chowk, Purani Abadi, Sri Ganganagar, Rajasthan, India

fauja.singh@live.in