TESTING AND VERIFICATION OF SOFTWARE MODEL THROUGH FORMAL SEMANTICS: A SYSTEMATIC REVIEW

Deepak Arora¹, Bramah Hazela²

¹Professor, Computer Science & Engg, Amity University, Lucknow
²Assistant Professor, Computer Science & Engg, Amity University, Lucknow

Abstract

UML is a standard language used in business modeling for specifying, visualizing and constructing artifact for software and non software systems. It provides the capability to explore the static structure as well as dynamic behaviour of any large and complex software system. It consists of different software design patterns, templates and frameworks with unique diagrams to represent different aspect of software design during its development phase. Model based verification has been a key area to be explored to establish the model consistency and validation formalization. Through massive survey it is found that still the literature is lacking the well formed rules and semantics for UML model verification at the early stages of any software development. This research work emphasises the development of novel techniques for the verification and validation of different UML models. It also focus on automated test case generations using formal semantics based on different pre-established mathematical theories related to graphs. Testing of any software can be broadly classified into three parts: test case generation, test execution and evaluation. Various tools and techniques have already been proposed by many researchers for automation of model verification specifically for object oriented software designs. In this paper authors have summarized and analyzed different approaches and methodologies related to automated verification of UML models and formalization of rules and semantics in order to automate the test case generation and its evaluation.

Keywords: UML, Formalization, Automatic Test Case Generation, Model Based Verification and Testing.

INTRODUCTION

Software modelling can be best described by the use of formal semantics. Selection of a proper model is the basis of modelling. There are various methods for modelling a software system that includes Unified modelling language (UML), Graph Transformation system (GTS), Abstract State Machine (ASM), State Diagrams. Unified modelling language provides broader range of language construct specifications to be constructed. A key areas in successfully using UML2 is understanding the semantics of the augmented language construct. For example, what is the meaning of a class diagram in terms of a component diagram, or what is the meaning of a state machine in terms of an activity diagram. Is there any approach for transformation of UML behavioural diagrams that will help in software testing and verification? These are not easy questions to answer and involve understanding the semantics of each individual construct.

All phases of software development starting from requirements analysis, design, implementation is discuss and maintain by UML diagram. The main goal is to model the software system before you build it. UML 2 specification defines two major kinds of UML diagram: structural diagrams and behavioural diagrams. Structure diagrams show the static structure of the system and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a software system, and may include abstract, real world and implementation concepts. Behavioural diagrams show the dynamic aspect of the objects in a system, which can be described as a series of changes to the system over time.

UML models are important source of information for design of test cases and verification. Model-based test case generation can be planned at an early stage of the software development life cycle, allowing to carry out coding and testing in parallel. Software testing is an important method to assure software quality. Traditional testing method based on handwork is low efficiency and cause the increase of test cost and time. With the development of testing technology, it advances high demand for to automate software system. The automated test cases generation is viewed as a guarantee to carry out effective and maintainable software testing. Model-based test case generation and verification technique becomes an obvious choice in software industries and is the focus of this work.

This survey aims at summarizing the automated testing and verification by covering the research queries below. The research queries aims at finding the efficient procedures for automated testing and verification of software model in practice. The queries are:

- RQ1- What are the various approaches for transformation of UML diagrams to other graphical structure?
- RQ2-Which is the most widely used technique?
- RQ3-What are the broad areas covered by these transformation techniques?
- RQ4- What is the need of formalization of UML diagrams?
- RQ5- What are the benefits of using various techniques?
2. SOURCES OF INFORMATION

This paper presents a systematic review of the work done in the field of automatic generation of test case particularly related to UML based automated test case generation and verification of software system. In order to gain a broader perspective, various papers and journals were searched. The following six databases were covered:
II. IEEE Xplore (www.ieeexplore.ieee.org).
III. Springer LNCS (www.springer.com/lncs)
IV. Science Direct (www.sciencedirect.com).
V. Journal of Object technology (www.jot.fm)
VI. Google Scholar (www.googlescholar.com)

3. SEARCH CRITERIA

The initial search criteria was kept broad in order to include the articles with different uses of terminology. The key words used were <transformation> and (UML or <UML Diagrams>) and <finite state machine> and <generation of grammar> and <software testing and verification>, and the database fields of title and abstract were searched. The start year was set to 1990 to ensure that most relevant research within the field would be included, and the last date for inclusion is publications within 2014.

The ultimate goal of software testing is to help designers, developers, and managers construct systems with high quality. Thus, research and development on testing aim at efficiently performing effective testing to find more errors in requirement, design and implementation. Progress toward this destination requires fundamental research, and the creation, refinement, extension, and popularization of better methods. The evolution of definition and targets of software testing has directed the research on testing and verification techniques.

4. DATA COLLECTION

The list of journals and conference proceedings with no. of paper referred is given below,
I. IEEE Transactions of Software Engineering: 14
II. Journal of system and software: 02
III. Software testing verification and reliability: 03
IV. ACM computing surveys: 05
V. IBM system of journals: 01
VI. IEEE Computer and application software: 05
VII. Computer and Information sciences: 04
VIII. Journal of object technology: 01
IX. International workshop of automation on software test: 02
X. IEEE conference on software maintenance: 06
XI. Proceedings of International conference on UML: 06
XII. Software engineering & Advanced applications: 02
XIII. Software reliability engineering: 02
XIV. ACM SIGSOFT software engineering: 07
XV. Computer science and Information technology: 03
XVI. Others: 06

The Unified Modeling Language (UML) [47] is a visual modeling language that comprises fourteen types of diagram representations to show structural and behavioural characteristics of any system. Now a days, there are many studies that are focused on test cases generation from UML specification and can be found in [20, 22, 25 26].

5. RESULTS

The following section reflects the results related to the research question.

5.1 RQ1- What are the Various Approaches for Transformation of UML Diagrams to other Graphical Structure?

There are various approaches for transformation of UML diagrams to FSM (Finite state machine), Abstract state machine (ASM) and other graphical structure which would be helpful for further verification of software system.

5.2 RQ2: Which is the Most Widely used Technique?

The most widely used techniques involve Model based software testing. One of the oldest approaches for model based testing is by using Use Case, class and State diagram. In these approaches, the models are transformed into its equivalent usage models to describe behaviour and usage of software system. Kansomkeat [41] proposed an approach using only state chart diagrams. The main advantage of this approach was the capability of automation.

5.3 RQ3- What are the Broad Areas Covered by these Transformation Techniques?

The broad areas covered by these techniques includes Compiler construction tool, real time embedded systems, artificial intelligence planning, spread sheets, OO systems, SOA interacting services.

5.4 RQ4- What is the need of Formalization of UML Diagrams?

Formalization of UML has become a prominent domain of research for the last few years. In this research query we will discuss a few works done in this domain related to formalization of UML static and dynamic models.

In all research works, UML diagrams have been Formalized using other formal languages. A context free grammar can be generated for widely used UML diagrams that are used in the design phase of the software life cycle namely Class, Sequence and State Chart diagram. In [69], we propose a Context Free Grammar for the analysis phase to establish traceability of requirements and consistency verification of UML Use case, Activity and Class Diagram. A UML compiler has been proposed in [68] that is a framework for syntactic and semantic verification of UML diagram. This
work proposes a grammar for Class and Sequence diagram. Here, we have taken into consideration the State Chart diagram also because the State Chart diagram depicts the state change of an object at runtime. We can also formalized other behaviour UML diagrams.

5.5 RQ5- What are the Benefits of using Various Techniques?

Each technique has its own advantages and disadvantages. A UML State chart [39] covers various test criteria such as transition coverage, full predicate coverage, transition pair coverage and complete sequence coverage. It also helps in performing class level testing. Activity Diagrams on the other hand can represent both conditional and parallel activities. A fork construct is used for concurrent activities in activity diagram. Sequence diagrams [44] describe sequence of actions that generate in a system over time. It captures invocation of methods from each object and order in which it occurs. Collaboration Diagrams [67] covers the dynamic aspect of testing better than any other UML model. Therefore, it can easily represent dynamic behaviour of the system along with good graphical representation of system scope requirements.

6. CONCLUSIONS

Unified Modeling Language (UML) has now become a de facto standard in the field of software testing and verification. New formalization techniques for the generation of test case from these UML diagrams need to be explored.

The overall objective of the study was to gather sufficient data to understand the nature of the various testing techniques available. These techniques will be more effective with formalized UML diagram.

The number of techniques proposed for test case generation is very large. We need to better understand the difference between these techniques and explore new methods for further improvements.

FUTURE WORK

Hence, the next step in this field of research will involve surveying and finding new possibilities in this area. Further, possibility of automation in test case generation via UML diagrams will also be explored simultaneously. Other possibilities include the improvements in testing techniques from UML diagrams. Further possibility is to automate test case generation and to explore other methods of using formal methods in software testing and verification.

REFERENCES

[7]. Memon, A.M. Pollack, M.E. Soffa, M.L (1999) Using a goal-driven approach to generate test cases for GUIs, International Conference on Software Engineering
[13]. Wei-Tek Tsai, Yinong Chen (2005) WSDL-Based Automatic Test Case Generation for Web Services Testing, IEEE International Workshop on Service-Oriented System Engineering
[17]. Hoffman, D; Strooper, P., ClassBench (1996-2003) A methodology and framework for automated class testing, SVRC, University of Queensland; Software-Practice & Experience, Technical Report
European Conference on Object-Oriented Programming, Springer Verlag, LNCS 1850, pp. 472-491


[23]. Valdivino Santiago1, Ana Silvia Martins do Amaral1, N. L. Vijaykumar (2008) QSEE project


[39]. Philip Samuel Rajib Mall (2009) Slicing-Based Test Case Generation from UML Activity Diagrams, ACM SIGSOFT Software Engineering Notes, Volume 34 Number 6, Page 1


[43]. Binder R. V., Testing Object-Oriented System Models, Patterns, and Tools, Addison-Wesley, NY, 1999


Conference on engineering complex computer system navigating complexity, vol 34, pp 1050-4729