Test Case Generation from UML Activity Diagram: The User Perspective Approach

Henok Bekele
Save the Children, Addis Ababa, Ethiopia
henzena@gmail.com

Mesfin Kifle
Department of Computer Science, Addis Ababa University, Ethiopia
kiflemestir95@gmail.com

Abstract

Prior to the release of a software to the customer, user acceptance test (UAT) will be carried out. UAT is the phase of testing which is used to determine whether a system satisfies the system requirements from the user perspective. The goal of user perspective testing is to make sure that the system completely supports the day-to-day business scenarios. But this manual UAT process has many limitations.

Unified Modeling Language (UML) models have emerged as an industrial standard for modeling software systems due to the wide acceptance of UML modeling language. More recently, they are being used in the generation of usage scenarios and test case scenarios. UML activity diagrams have got significant attention in test case generation. They have also become an ideal UML diagram to model the business process and its related concepts at higher level.

In this paper, a new approach is proposed to enable the end-user to automatically generate user level test cases from the UML Activity Diagram. The approach basically employs a series of transformations and a newly developed algorithm to get the user level test cases from the Activity Diagram. The Approach uses all path coverage criteria to generate the test cases.

A case study was conducted on point of sales (POS) system users to collect user perspective criteria to test the software system. The result of the case study has indicated that user perspective testing should focus on assessing if the system can support the day-to-day business scenarios. A prototype tool is also developed that validates the proposed approach.

Keywords: User Level Test Case; End-users; User Activity Graph; Transition List; User Activity Path Transition List

1. Introduction

Software Testing is an important phase in the software development life cycle that has to be managed very well as it plays a crucial role in assuring software quality [1]. The different phases of a software life-cycle involve different groups of stakeholders whose knowledge and understanding of software engineering differ heavily. For example, the software developer or architect has advanced knowledge about the software engineering and represents the system on his/her own way from the requirements of the end-users. On the other hand, the end-user has a deep knowledge of the domain with little or no understanding of software engineering concepts. Accordingly, at the time of testing, the system developers and end-users will have their own perspective to test the delivered system.

When end-users test the system from their point of view, they are mostly interested in testing the functionality of a system based on the acceptance or user perspective testing criteria as documented in a contract by focusing solely on the outputs generated in response to their input, ignoring the internal working mechanism of a system. In other words, they focus on checking the behavior of the system under different scenarios which they face every day in real world while doing the same task manually. This is purely a black box testing which we would like to concentrate in this research.

Sometimes, system developers involve end-users in testing the delivered system on user acceptance testing stage but this manual process has many

problems. For example, it depends on the quality of the people involved, mostly prone to error, consumes huge amount of time, and may not generate enough test cases to test the system effectively. End-users can be also influenced by the system developers in the process of producing test cases.

So in this research, we propose an approach to enable end-users to automatically generate user level test cases from the system model, particularly from the UML Activity Diagram based on user perspective criteria with minimum assistance from the system developer.

The rest of the paper is organized as follows: Section 2 discuses the background information on the basic ideas of the paper. The third section describes a case study which is carried out to gather the user perspective criteria from end-users and the fourth section discusses the proposed approach for generating test case from UML Activity Diagram focusing on the end-user perspective. The fifth section describes the implementation of a prototype. The sixth section assesses related works on the area of test case generation from UML Diagrams and finally section seven concludes the paper.

2. Background

Development of a reliable software system has to be approached in a systematic way. This requires use of appropriate tools and mechanisms to ensure a high level of quality for the system under development. Software testing is a process of verifying and validating that a software application or program meets the business and requirements that guided its design and development [3].

There are many types of software testing techniques but they can be defined in three dimensions. One dimension is showing the scale of System Under Test (SUT), which ranges from a small unit up to a whole system. The second dimension shows the different characteristics that we may want to test. The third dimension shows the kind of information we may want to use during software testing [2].The testing types under each dimension are not totally exclusive and there are also many other types of testing which are based on these fundamental testing types. In this research, we have focused on User Perspective Testing.

Software testing is usually done by the system developers but end-users have to test the system from their view before they use the delivered software in live working environment, which can be referred to as user perspective testing. The User Perspective Testing aims at verifying and validating the implemented system in the real world by the end-user according to the original needs [4]. In other words, the goal of software testing from the system developer perspective is “to make sure the software meets the specification” or “to make sure that the developed software is bug free”. Whereas the goal from the user perspective testing is “to make sure that the system completely supports the day to day business scenarios along with other known possible scenarios that may create hurdle in business operations, and to make sure that the software will not hurt the LIVE environment ” [5].

User Acceptance Test (UAT) can be considered as a typical example of User Perspective Testing. UAT is the phase of testing which is used to determine whether a system satisfies the requirements specified in the requirements analysis phase. The acceptance test design is derived from the requirements document which describes the system’s functionality, interface, performance, data, security, etc.

In Model Based Testing (MBT), software architects use UML models to represent a system. These models are used in a software specification document which is a baseline for the subsequent software development process including software testing. Normally both the developer and the end-user should test the system to make sure that the system works in accordance with the requirements. Mostly in MBT, when the system developers test the system, they use the test cases generated automatically from the UML models. On the other hand, the end-users want the test cases generated from the models to be high level so that they can understand them clearly at the time of testing. So the generated test cases can be referred to as user level test cases because they can be used by end-users to

test a software system with possible minimum support from the system developer.

UML Activity Diagram has been used more frequently to model the high level business process as it requires minimal effort and time to develop and understand. It also represents the business flow in more abstract form than other UML Diagrams [6, 7, 8].

3. Case Study

The main purpose of this case study is to gather the user perspective criteria to test a system from the actual end-users. In the case study, both qualitative and quantitative approaches have been used. We have focused on the users of a Point of Sales (POS) system and a systematic random sampling technique is used to select representative POS system user companies. An interview is also used as a data collection method and finally the primary data collected by interviewing the end-users of representative POS users has been organized and analyzed.

According to the quantitative analysis

- 85% of the respondents think that the software system should be tested by the end-users before these users start using it in live working environment.
- Only 10% of the respondents have tried to conduct an informal test to check the system works as intended by themselves.
- Almost 90% of them think that conducting a software test from the user perspective would make them very comfortable on live environment.
- 80% the respondents think that if enough training is given on the system, it is better that the system developer doesn't involve in the actual end-user test process.
- Only 5% of the end-users are fairly encouraged to test the system from their perspective.

The qualitative analysis has also showed that the majority of the respondents wanted to test the system from their point of view and they also mentioned that user perspective testing should focus on checking whether the system supports the day to day business scenarios. But mostly they can't practically perform this type of test because: there is no established mechanism to perform the test, luck of software knowledge to adopt the system developer approach to test the system, and the normal manual process would take a lot of time.

4. The Proposed Solution

In this approach, UML activity diagrams are used to generate user level test cases for testing a software system from the end-user perspective. The approach basically employs a series of transformations to get the final user level test cases from the specific Activity Diagram. Below, the basic steps of the proposed approach are listed.

Step 1: Transform the Activity Diagram into an intermediate representation.

- Export the Activity Diagram to XMI file.
- Parse the XMI file to get the required Model Information.
- Analyze the Model Information and transform it into a Transition List (TL).

Step 2: Develop a new algorithm to build a User Activity Graph (UAG).

Step 3: Extract all paths from the resulting UAG by applying all path coverage criteria. The resulting paths will be analyzed and used as test cases.

In the first step, the activity diagram will be transformed to an intermediate representation which contains only related information that is used in the process of test case generation from the user perspective. The intermediate representation is needed because even though the activity diagram presents the system flow in high level compared with other UML activity diagrams, it still contains detailed information for the purpose of generating test cases from the user perspective.

In this step, the activity diagram is firstly converted into an XML Metadata Interchange (XMI) file which is the OMG standard for exchanging metadata information via Extensible Markup Language (XML). In the next stage, the XMI file will be parsed to get the intended model information because the XMI file contains other extra details to redraw the UML Model diagrams in addition to the

model information. In the last stage, node based analysis will be carried out on the parsed model information to get the final intermediate representation which only contains a transition list that connects user related activity nodes. This is achieved by removing unnecessary nodes from the model information without affecting the original business flow of the system.

In the second step, User Activity Graph (UAG) is constructed by taking the transition lists as input from the first step. UAG is basically a cycle free directed graph. To get the intended cycle free UAG, an algorithm is developed which assumes any cyclic transitions or transition which causes cycles shouldn't occur more than once along in a given path in a UAG. In other words, the algorithm first identifies the cyclic transition lists from the global transition lists TL by analyzing the relationship of the nodes in the TL and it will then construct the UAG by adding the transitions and the respective nodes from the TL but in the process it avoids adding cyclic transitions if it has already appeared in a particular path in UAG to eliminate the formation of infinite loop in UAG. The algorithm maintains UAG Transition List (UAGTL) to internally represent the UAG that contains the depth of the UAG, unique self and parent identification per depth. UAGTL is essential for two purposes. Firstly, it helps to back track a given path in UAG to check the presence of cyclic transition at the time of UAG construction. Secondly, a user activity path can easily be extracted by following a given path in UAG to generate test cases.

In the third step, all the paths will be extracted from UAG by traversing UAG using modified depth first search or alternatively from UAGTL which was developed at the time of UAG construction. This can be achieved by checking the presence of the leave nodes and following the parent to child information all the way to the root node and repeating this process for all leaf nodes at all depths of a UAG. These generated paths are considered to have all the possible scenarios of the system as they are directly derived from the activity diagram which again contains user requirement information from the user requirement analysis phase. End-users can follow these paths to check whether the system behaves as intended at the time of testing. End-users are also provided with decision information along with a particular path so that they can identify which scenario they are following. Having this information from the tool, end-users can be able to test the system from their perspective with minimum involvement from the system developer.

5. Prototype

A prototype is also developed to demonstrate and validate the new approach. This prototype performs all the transformation steps that are mentioned in the approach and generate user level test cases. It will be used by end-users to test the system from their perspective.

The prototype is window based system which is developed using Microsoft Visual Studio 2010 and C Sharp Programming Language. Visual Studio 2010 provides a .net framework 4 that incorporates LINQ which is an efficient tool to parse the XMI file. SQL Server 2008 database is also used in the implementation to store the intermediate representation at different stages of the system process.

This prototype is designed to take input from UML modeling tools in XMI format. For this purpose, we have used Altova UModel 2011 which can model the business flow of the POS system using Activity Diagram and export the model information in XMI format. The tool will take the exported XMI file and perform two consecutive transformations, i.e., parsing the XMI file using Microsoft LINQ and Node based analysis of the model information which results in Transition Lists that contain only user related activities by removing unnecessary nodes and maintaining the original business flow. The tools will then construct UAG by applying the algorithm on the Transition List from the pervious stage and maintain the UAG transition list which serves as an internal representation of the UAG. Finally, it generates user level test cases from the UAG. Each test case contains user activity path or sequential user activities along with a set of guarding conditions or scenarios as shown in Table 1. The end-users can test the system by carrying out the sequential user related

activates and comparing it with the associated guarding conditions and make sure the system behaves as stated in the user acceptance criteria.

Table 1: Sample Test Cases for Refund Activity Diagram.

TCN	Actor	User Activity Path	Guarding Condition
1	Casher	SN→STM→SDR→SMC→LIT →SSI→FRD→RSI→PRC→FN	• Query need additional criteria • Query returns result , • Valid refund detail is entered
2	Casher	SN→STM→SDR→SMC→LIT→SSI→ FRD→RSI→FRD→RSI→PRC→FN	• Query need additional criteria • Query returns result , • Invalid refund detail is entered
3	Casher	SN→STM→SDR→SMC→LIT→SDR→LIT →SSI→FRD→RSI→PRC→FN	• Query need additional criteria • Query returns no result , • Valid refund detail is entered
4	Casher	SN→STM→SDR→SMC→LIT→SDR→LIT →SSI→FRD→RSI→FRD→RSI→PRC→FN	• Query need additional criteria • Query returns no result , • Invalid refund detail is entered
5	Casher	SN→STM→SDR→SMC→LIT→SDR→ SMC→LIT→SSI→FRD→RSI→PRC→FN	• Query need additional criteria • Query returns no result , • Query need additional criteria • Valid refund detail is entered
6	Casher	SN→STM→SDR→SMC→LIT→SDR→ SMC→LIT→SSI→FRD→RSI→FRD→RSI→PRC→FN	• Query need additional criteria • Query returns no result , • Query need additional criteria • Invalid refund detail is entered

6. Related Work

Due to the popularization of UML modeling techniques, it has emerged as the de facto standard for modeling software systems. UML can be used to describe different aspects of a system including static, dynamic, and use case views of a system. There are many researchers who propose different approaches of automatic test case generation from different UML diagrams. But UML diagrams such as Activity Diagram, State chart and Sequence diagram have got significant attention by many researchers [1, 9, 10, 11, 12]. The researchers have selected specific UML tools based on their objective and motive of their research. For example, the authors in [1, 10] used Activity Diagram to generate test cases which is used to test a system on cluster or system level whereas authors in [9] used state chart diagram to achieve component level testing. On the other hand, the authors in [11, 13] took the advantage of more than one UML diagram for the purpose of generating test cases which have adequate test coverage criteria to test the system. Below, we have tried to briefly explain some of them that are closely related to our work.

Kundu and Samanta [10] have presented an approach of generating test cases from activity diagrams using UML syntax and with use case scope. They considered a test coverage criterion, called *activity path coverage* criterion, which considers both loop testing and concurrency among activities of an activity diagram. But the approach they followed to test the system is purely from the system developer context.

Ieamsaard and Limpiyakorn [4] have presented a method and a system implemented to generate user acceptance tests from use case descriptions once the requirements analysis phase has been committed. The paper is relevant to our work as it focuses on the area of user acceptance testing but test cases are generated from the use case description and finite sate machine is used to generate a test path graph instead of UML activity diagram. It is not also clearly stated that the approach considers only user relevant processes in the course of generating test cases.

Kumar *et el*. [13] have proposed an integrated approach to generate test cases from UML sequence and activity diagrams. They first transform UML diagrams into a graph. Then, they developed an algorithm to generate test scenarios from the constructed graph. Next, the necessary information for test case generation, such as method-activity sequence, associated objects, and constraint conditions, are extracted from test scenario. In this approach, they used sequence diagram in addition to UML activity so that it also considers the message

sequence of objects in the system but this internal communication of objects is not relevant for the purpose of test case generation from the user perspective.

In general, as can be seen from the reviewed articles in the area of automatic test case generation in model based testing, the process of test case generation, whether from the developer or end-user view, can be influenced by the method of modeling the business process of the software system, the intermediate representation of the model to drive the test case generation, and finally the test coverage criteria. From the review, we can easily see that most of the researchers have focused on generating test cases from the developer perspective which as a result, the test case generation mechanism has shown a significant improvement in the past. But on the other hand, the automatic user level test case generation has been given less attention. As a result, the works that have been done in this area are almost in their infant stage.

7. Conclusion

In this paper, a new approach is presented to generate user level test cases from a UML Activity Diagram. The major significance of this approach is to give the end-user the capability to test the software independently of the software developers. In this approach, the UML Activity Diagram, which models the business flow of the system, was first translated in to an XMI file and then a transition list has been developed by parsing the XMI file and analyzing the resulting parsed model information. An algorithm has also been developed to construct the UAG which is loop free directed graph that contains only user related activities and finally user level test case has been generated from the UAG that would be used by end-users to test the system.

References

[1] Xiaoqing Bai, C. Peng Lam, and Huaizhong Li, “An Approach to Generate the Thin-threads from the UML Diagrams”, Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC’04), 2004.

[2] Yasir Masood Malik, “Model Based Testing: An Evaluation”, Blekinge Institute of Technology, May 2010.

[3] John E. Bentley, Wachovia Bank, and Charlotte NC, “Software Testing Fundamentals Concepts, Roles, and Terminology”, SUGI 30, 2005.

[4] Chavalid Ieamsaard and Yachai Limpiyakorn, “On Integrating User Acceptance Tests Generation to Requirements Management”, International Conference on Information Communication and Management, IACSIT Press, Singapore, 2011.

[5] Abubukar Manuawar, “How to User Acceptance Test (UAT) Tips & Templete for Software Business Analyst”, Pakistan, 2010.

[6] D. Pilone and N. Pitman, “UML 2.0 in a Nutshell”, O'Reilly, June 2005.

[7] Nick Russell, Wilm P. van der Aalst, Arthur H.M. ter Hofstede, and Petia Wohed, “On the Suitability of UML 2.0 Activity Diagrams for Business Process Modeling”, In 3rd Asia Pacific Conference on Conceptual Modelling (APCCM’06), Hobart, Australia, 2006.

[8] Stuart Reid, “Systematic UML testing”, In Proceedings of the Software Quality Systems Conference 2005, Westminster, London, 2005.

[9] G. Booch, J. Rumbaugh, and I. Jacobson. “The Unified Modeling Language, Reference Manual”, Addison-Wesley, Reading, Massachusetts, 1999.

[10] Debasish Kundu and Debasis Samanta, "A Novel Approach to Generate Test Cases from UML Activity Diagrams", in Journal of Object Technology, Vol. 8, No. 3, May 2009, pp. 65-83.

[11] Santosh Kumar Swain, Durga Prasad Mohapatra, and Rajib Mall, “Test Case Generation Based on State and Activity Models”, Journal of Object Technology, 2010.

[12] P. Fröhlich and J. Link, “Automated Test Case Generation from Dynamic Models”, ECOOP 2000, Lecture Notes in Computer Science, Jan 2000, pp. 472–491.

[13] Santosh Kumar Swain and Durga Prasad Mohapatra, “Test Case Generation from Behavioral UML Models”, International Journal of Computer Applications, Vol. 6, No. 8, September 2010.