Test case generation using UML activity diagram - A survey

Ms. Hetal J. Thanki,Prof. S.M.Shinde
#1 Department of computer engineering, JSCOE, Hadapsar, Pune. University of pune, 9765725551
#2 Department of computer engineering, JSCOE, Hadapsar, Pune. University of pune,8796523794

ABSTRACT

Test driven design (TDD) and design driven testing (DDT) are used for test case generation. TDD generates so many duplicated test cases at the end of the project. DDT is novel approach to generate test cases based on design model of application. Comparative study indicates batter result using DDT. White-box traditional regression testing depends on analysis of impact of changes in source code. This practice minimizes the amount of testing required to validate code changes but they do not influence on requirement specification. Black-box testing supports ability to test from higher level design and requirement. Test optimal is a tool to generate test cases based on model driven environment using UML activity diagrams. Traditional approach generates many numbers of large and duplicate test cases. This paper focuses on test case generation techniques and describes proposed approach for improvement in test case minimization and prioritization. Proposed approach will minimize and prioritize generated test cases and generate optimal test cases using model driven testing. Changes in modified activity diagram element will identify Reusable and retestable test case. Filtering and prioritization will lead to better available resource utilization and will improve software project management.

Key words: Activity diagram, Coverage criteria, Model driven testing, Regression testing, Test prioritization.

INTRODUCTION

With the advent of technology software becomes very crucial part in all the institute and industries. To develop a software and to test it as per customer requirement is very important because the software which is not able to satisfy customer requirement after development will leads to increase and waste of cost, time and effort of all parts of organization. It is batter to start testing process as early as possible in software development process. Delaying testing after implementation and development will lead to high risk and if bugs are found at latter stage than schedule of project often slip. Plan testing at early stage of software development lifecycle will help to increase coverage, software quality and efficiency.

For software testing which consists of many interrelated tasks, each with its own artifacts and deliverables creation of test cases is the first fundamental step. Then test procedures are designed for these test cases, and finally, test scripts are created to implement the procedures. Test cases are key to the process because they identify and communicate the conditions that will be implemented in test and are necessary to verify successful and acceptable implementation of the product requirements. They are all about making sure that the product fulfills the requirements of the system[6].

Test driven design (TDD) and design driven testing (DTT) are the two software development and test case generation methodology, which are used for test case generation. TDD generates so many duplicated test cases at the end of the project. DDT is novel approach to generate test cases based on design model of application. Comparative study indicates batter result using DDT. Test optimal is a tool to generate test cases based on model driven environment using UML activity diagrams or state diagrams. Traditional approach will generate many numbers of large and duplicate test cases.

MATERIALS AND METHODS

UML activity diagram

Introduction

"The Unified Modeling Language (UML) is a language for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems". "UML is a graphical notation for modeling various aspects of software systems." UML is released by object management group (OMG) in 1997. UML is adapted as common and standard design language to develop software application. UML is developed by Grady Booch, James Rumbaugh and IvarJacsobson. UML is programming language independent.

There are different types of UML diagram available for system design like class diagram, package diagram, composite structure, component diagram, Deployment diagram, Use case diagram, Statechart diagram, Activity diagram, sequential diagram, Interation diagram, etc.

Activity diagram depicts flow of behavior of a system that extracts the centre idea from flowchart. It applies to number of domain like software modeling, business processes, software processes and workflows. The activity diagram contains activity states that are made up of smaller actions which are representedin the implementation of a statement in a process or the performance of an activity in a workflow [6][13]. Activities are intended to be reused within an application and actions are typically specific and are used only within a particular activity [13]. The notations are inspired by flowchart and state transition graphs [6]. Concurrent behavior, loop, behavior of use case and event driven behavior of complete system can beimplemented by activity diagram.

Activity diagram starts with initial node and end with final node. Activity can be written in rectangle or box with round corner. Detail code, pre condition and post condition for activity and actioncan be included with activity diagram. Notes for each actions can be specified through dog eared rectangle box.

The nodes represent processes or process control including action states, activity states, decisions, swim lanes, fork, join, objects, signal senders and receivers [6]. Flow through activity and actions of activity are link through activity edges. It specifies control and data flow between

oneaction to next. Activity edges represented by arrow line with optional name of edge. Some kind of filtering like select those employeeswhose salary is greater than 5000 is also specified in activity diagram. Some time action required more than one valid input for execution. That are specified at input pin in group through parameter set. Parameter set is shown by drawing a rectangle around an activity box. Same like input parameter set one can specify output parameter set with output pins.

In activity diagram transitions are shown as arrows. Branches are shown as diamond with one incoming arrow and multiple exit arrows. Each arrow may be labeled with Boolean expression to be satisfied to choose the branch [6]. Forks are represented by multiple arrows entering and joins are represented by multiple arrows leaving a synchronization bar. Activity edge may have weight specifies number of input (data, object etc.) required to perform target action. Weight is specified through curly brackets (Weight=9 requires 9 inputs to execute target action). Guard condition can also be specified with decision symbol by putting Boolean expression in brackets ([ans=yes or and=no]).

Activity nodes are of three types that are first parameter node to represent data being passed, object node for complex data and control nodes to direct the flow through an activity diagram.

Nodes of activity diagram

1. Initial node

It is a starting point of activity diagram with no incoming edge and represented by solid black dot.

2. Decision node

It is a control node that chooses different output flows based on Boolean expression. It has one incoming flow and multiple outgoing flows, of which only one will be taken. Guard condition should be made to ensure that only one flow can be taken [6]. It is represented by diamond symbol.

3. Merge node

It is opposite to a decision node. It brings together multiple alternative flows and from that chooses one among them as output flow.

4. Fork node

It splits an incoming flow into multiple concurrent outgoing flows. Token or data arriving at a fork are duplicated across each outgoing flow.

5. Join node

It is opposite to fork that synchronizes multiple flows of an activity back to a single flow. All tokenand data of incoming edges are send over the outgoing edge.

6. Final nodes

Final nodes are of two types: Activity final and flow final. For termination of entire activity, activity final node is used and for termination of particular path of an activity diagram but not entire activity. Activity final is indicated by black dot with a circle around it. Flow final is indicated by a circle with X in it. Figure 1 Shows control node of activity diagram.

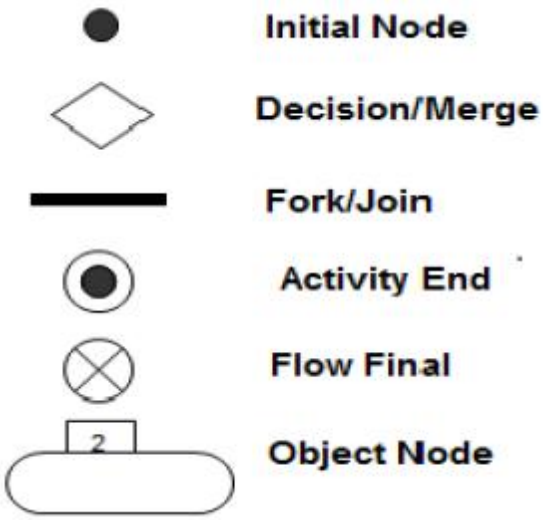

Fig 1. Control node of activity diagram.

Figure 2 shows general example of activity diagram including whole control node. Where A1 to A7 are activities or actions of activity diagram. Node A2 has input and output pin. Activities in nodes A1, A2,A4, A5 are concurrent activities. Activity path A4, A5 is terminated with flow final node. Activity A2 andA3 are join through join node as shown in figure 2. All the data or token of A2 and A3 are passed to A6. Detail activity diagram with data store, loop, streming actions, expantion region, buffer node can be drown [13].

This activity diagram describes expected behavior of an operation. Incorrect implementation of activity diagram results in unexpected behavior of an operation. Tools are available to draw and change or modify activity diagram.

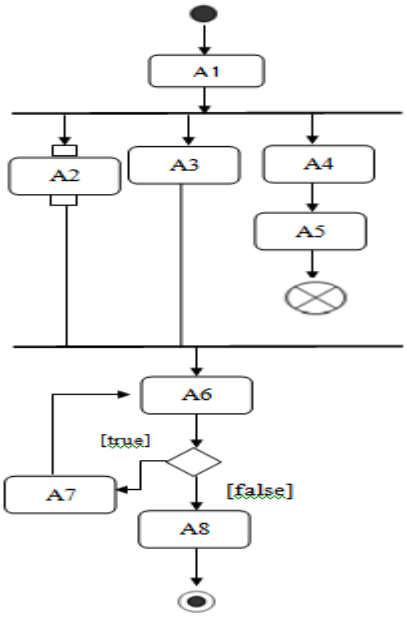

Fig 2: General Example of Activity diagram

Discussion

Many different techniques have been developed in past for test case generation with the help of different diagrams of unified modeling language. Authors of different papers have taken activity diagram or sequence diagram of UML and generate different graphs from that. Traversing the graph through depth first search or best first search will prove activity path for application. This activity path is further analyze with to reduce number of traces or path and test case generation using different coverage criteria like all path coverage, simple path coverage or basic path coverage. Activity path coverage criteria is also new approach. L.C. Briand et al. [1] have built a system for system level testing using various use case scenarios. In this paper authors have used sequence diagram to represent various use case scenarios. After that they collect various test information, test requirements, test cases from the sequence diagram using object oriented language (OCL). J. Hartmann et al. [2] have considered UML activity diagrams. In this paper textual description of use case is taken as input and then this textual description is automatically converted into activity diagram. After that new test cases are generated using analysis of activity diagram. X. Bai et al.[3] scenario based testing is considered and activity diagrams are arranged in hierarchies such that top level activity diagram capture dependency among use cases and low level activity diagram represent behavior of the use cases. This structure of activity diagram is than convert into graph then sequence of execution is used for test case generation. Mingsong et al. [4] Test cases are generated from high level design models which represents expected structure and behavior of software. Test cases are generated based on basic path coverage criteria. Chen et al. [5] Authors have considered random generation of test cases for java programs and test cases are reduce by comparing simple path with program execution traces. puneet et al. [6] authors have compared two techniques of authors kundu and samanta (2009) [9] and yasir [12] and present graphical analysis. Kim H. et al. [7] proposed a method based on I/O explicit activity diagram abstraction obtain from the fully expanded activity diagram by exposing only external inputs and outputs and by converting it to accept event and change signal. This method is for the system which contains concurrently executing objects. Through IOAD model directed graph is constructed to extract test scenarios and test cases using all path test coverage criteria. It is based on concurrently executing object system solution or test case generation. Linzhang W et al. [8] have developed a prototype tool called UMLTGF. It uses gray box method to generate test cases and design is reduce to avoid test model creation. Kundu and samanta [9] have taken activity graph as an input and generate output as activity path. They uses UML 2.0 and generate test suit through activity path coverage criteria which covers synchronization faults, loop faults etc. Roberto S. et al. [10] have extend TDE/UML is a tool to generate test cases based on model driven environment using UML diagrams. Traditional regression testing procedure is bottom- up that depends on changes in source code. This approach and model driven testing supports to generate and prioritize test cases based on top-down testing approach. This approach generates test cases based on user defined concerns and depends on trace-ability link between models, test cases, code and also on user defined properties associated with model elements. It extends model based testing environment. In paper [11] authors have used sequence diagram of UML as input and generate test cases for mobile application feature testing. They convert UML sequence diagram to labeled transition system and then by traversing LTS through DFS it generate coverage path. With this all coverage criteria path they generate test cases.

Different model driven testing techniques and its comparison with different criteria like modeling language used, tool support, the testing targets etc. given in [20]. Testing of

applications of different domains is important and crucial task. Use of Web application increases day by day. To test web application demand of systematic methodology is increasing. Framework for supporting such methodology which is loosely coupled using different models is described the system under testing web application model is generated. Based on web application model, test case models are generated. To describe the environment and process of test execution test deployment and test control model are generated. The test engine executes test cases automatically and results are reflected to test case models [21]. In network domain network management interface model driven testing technique is developed [22]. In this method from platform independent model (PIM) and platform specific model (PSM) defined in network management interface specification, test case model and test scenario model constructed either automatically or manually. Interface testing of 3G mobile communication networks is done through implemented automatic testing platform tool. Model driven conformance method [23] is used for 3G network management north bound interface to cope with technological and specification changes. In this method interface technology independent test model (PIT) is transformed to interface specific test models (PSTs) which can be used in conformance testing by proper test tools. PIT is derived from platform independent model. For Collaborative Embedded System Design (for embedded domain) testing is key issue. It is difficult to design test cases for Security protocols for security critical application (security domain) SecureMDD is combined with design of functional and security test. By this method it is easy to define test cases during modeling stage [29]. It also generate runnable test for application. Case study is described for open source data structure using zellers algorithm for failing test case minimization which reduce length of sequences of method calls [24]. Based on analysis of relationship among testing requirement and test cases, software faults and changes, priority of testing requirements one method called regression test case design is developed which achieves the function of checking the regression test case suite [25]. To generate batter test suit including longer test cases which achieves higher fault detecting and higher coverage of code experimental studies in a scenario of specification based testing for reactive systems is developed but application of test case minimization will create opposite effect [26]. Automatic test cases in software product line [27] uses standard UML 2.0. Automatic test case generation based on model driven architecture is done through system model to test model conversion and test model to test code generation which is done through model to model conversion method and model to code conversion method respectively [15].

Proposed work

Many test case generation methods are used to generate test cases through different activity diagram. Our proposed approach uses TEST OPTIMAL TOOL to create activity diagram. From this activity diagram of a domain like finance proposed system will generate test cases. In regression testing changed software must be checked for its correctness. In traditional approach of regression testing generated test cases are more and may have duplication also. Our system divides generated test cases into reusable and retestable test cases. Tester has to focus first on retestable test cases because they are affected by change in activity diagram or desin of software. That retestable test cases are further prioritized through our system so that it will reduce testing time, effort and cost of software.

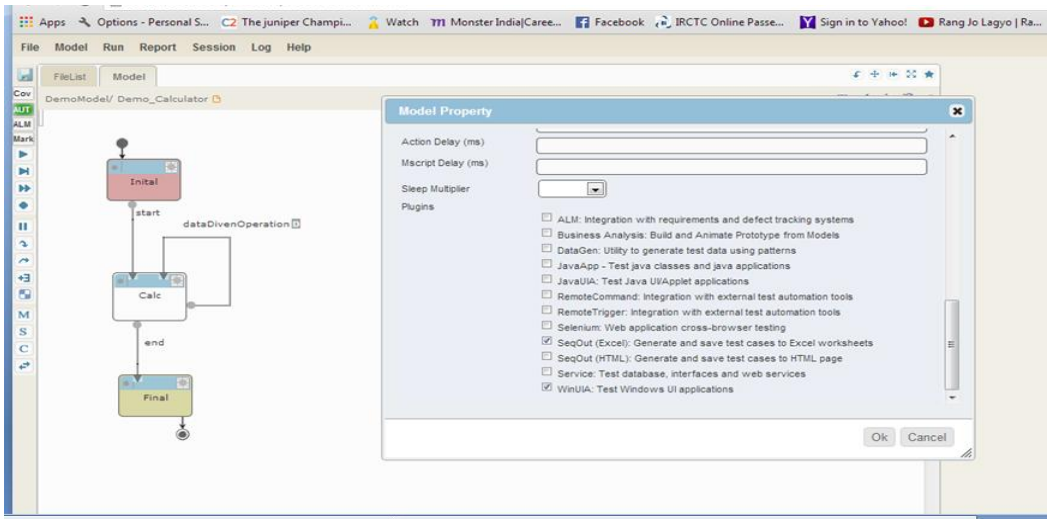

Fig 3. Test optimal tool

CONCLUSION

Software testing has been critical part of entire SDLC where success of software projects is heavily depends. Therefore there have been wide spread efforts to improve SDLC practices especially during testing. In our paper we present a model driven testing approach with improved minimization and prioritization of test cases. We believe that such prioritization of test cases will result in better utilization of available resources and improved software project management. Time complexity of proposed project is depends on number of input value (test cases). In future test cases can be prioritize by considering various domains for which software being developed and profile of users.

REFERENCE

[1] L.C. Briand and Y. Labiche.A UML-based approach to system testing. In 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, pp. 194-208,2001.

[2] J. Hartmann, M. Vieira, H. Foster, and A. Ruder.A UML-based approach to system testing. Innovations in Systems and Software Engineering, 1(1):12-24, April 2005.

[3] X. Bai, C. P. Lam, and H. Li. An approach to generate the thin-threads from the UML diagrams. In 28th Annual International Computer Software and Applications Conference (COMPSAC04), pp. 546-552, 2004.

[4] Mingsong, C., Xiaokang, Q., &Xuandong, L. (2006). Automatic Test Case Generation for UML Activity Diagrams. ACM, 2-8.

[5] Chen, M., Qiu, X., Xu, W., Wang, L., Zhao, J., & Li, X. (2009). UML Activity Diagram-Based Automatic Test Case Generation for Java Program. The Computer Journal, 52 (5), 545-556.

[6] Puneet E. Patel, Nitin N. Patil (2013). Testcases Formation using UML Activity Diagram. 2013 International Conference on Communication Systems and Network Technologies, IEEE 884-889

[7] Kim,H., Kang, S., Baik, J., &Ko, I. (2007). Test Cases Generating from UML Activity Diagrams. IEEE, 556-561.

[8] Linzhang, W., jiesong, Y., Xiaofeng, Y., Jun, H., Xuandong, L., &Guoliang, Z. (2004). Generating Test Cases from UML Activity Diagram based on Gray-Box Method. IEEE 284-291.

[9] Kundu, D., &Samanta, D. (2009). A Novel Approach to Generate Test Cases from UML Activity Diagrams. Journal of Object Technology, 8(3), 65-83.

[10] Roberto S. Silva Filho, Christof J. Budnik, William M. Hasling, Monica McKenna, Rajesh Subramanyan, Supporting Concern-Based Regression Testing and Prioritization in a Model-Driven Environment, 34th Annual IEEE Computer Software and Applications Conference Workshops, 2010 PP 323-328.

[11] Cartaxo, E., Neto, F., & Machado, P. (2007). Test Case Generation by means of UML Sequence Diagram and Labeled Transition System. IEEE, 1292-1297.

[12] YasirDawood Salman Almulham. Automatic Test Case Generation from UML Activity Diagram Using Activity Path. Malaysia 2010.

[13] D. Pilone and N. Pitman. UML 2.0 in a Nutshell.OReilly, June 2005.

[14] Test driven development, http://searchsoftwarequality.techtarget.com/de_nition/test-driven-Development

[15] http://www.agiledata.org/essays/tdd.html

[16] http://agilepainrelief.com/notesfromatooluser/2008/10/advantages-of-tdd.html

[17] IBM library, www.ibm.com/developerworks/rational/library/769.html

[18] UML, the Uni_ed Modeling Language William H. Mitchell (whm) Mitchell Software Engineering (.com), http://www.mitchellsoftwareengineering.com/IntroToUML.pdf

[19] Matt Stephens and Doug Rosenberg, Design Driven Testing, Test Smarter, Not Harder, Apress, 2010

[20] Mohamed Mussa, Samir Ouchani, Waseem Al Sammane, AbdelwahabHamou-Lhadj, "A Survey of Model-Driven Testing Techniques", 2009 Ninth International Conference on Quality Software, IEEE, pp 167-172, 2009

[21] Nuo Li, Qin-qin Ma, Ji Wu, Mao-zhong Jin, Chao Liu, "A Framework of Model-Driven Web Application Testing", Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06), IEEE, 2006

[22]Yinghui Chen, LiminHou, Yichang Liu, Yongping Zhang, Feng Qi, "Study and implementation of model driven testing method for network management interface", Proceedings of ICCTA2009, pp. 259-263, IEEE, 2009

[23] Liu Yi-chang, Chen Ying-hui, Qi Feng, QiuXue-song, "A Model-Driven Conformance Testing Method for 3G Network Management North Bound Interface", IEEE, pp. 323-326, 2010

[24] Yong Lei ; Andrews, J.H., "Minimization of randomized unit test cases", 16th IEEE International Symposium, pp. 267-276, IEEE

[25] WenhongLiu ;Xin Wu ; YuhengHao, "Research and Application of Regression Test Case Design Methods Based on the Analysis of the Relationship", Fifth International Conference, IEEE, pp. 233-236

[26] Fraser, G. ;Gargantini, A., "Experiments on the test case length in specification based test case generation", IEEE, pp. 18-26, 2009

[27] BeaBeatriz Pérez Lamancha, "Model-Driven Testing in Software Product Lines", Proc. ICSM 2009, Edmonton, Canada, pp. 511-514, IEEE

[28] Yang Liu ,Yafen Li, Pu Wang , "Design and Implementation of Automatic Generation of Test Cases Based on Model Driven Architecture", 2010 Second International Conference on Information Technology and Computer Science, 2010, pp. 344-347, IEEE

[29] KuzmanKatkalov, Nina Moebius, Kurt Stenzel, Marian Borek and Wolfgang Reif, Model-Driven Testing of Security Protocols with SecureMDD, IEEE, 2012

[30] Matt Stephens and Doug Rosenberg, Design Driven Testing, Test Smarter, Not Harder, Apress, 2010